PROJECTS ■ THEORY ■ APPLICATIONS ■ CIRCUITS ■ TECHNOLOGY

NUTS AND VOLTS

EVERYTHING FOR ELECTRONICS

www.nutsvolts.com
May 2015

- BASICS - How To Use A Transistor As A Switch
- Ham's Wireless Workbench - RF Oscillators
- Decode SW Radio Signals With Your PC
- PICAXE to PC Serial Communication
- Bring Vintage Radios Back To Life

Build this DIY RETRO RADIO

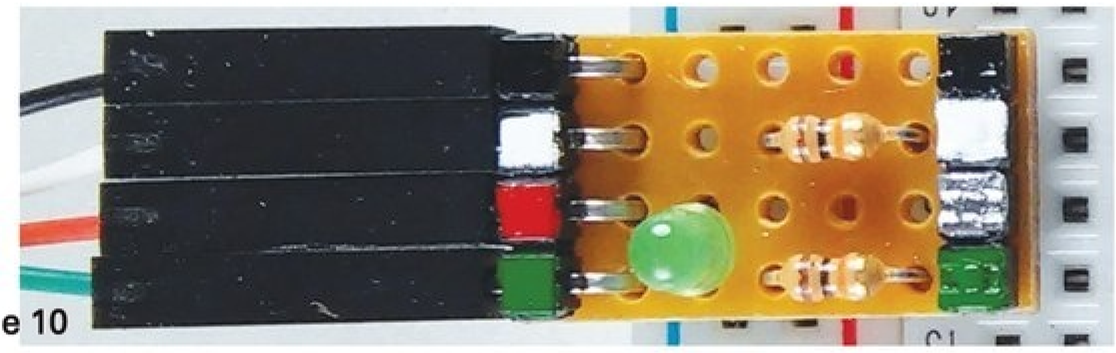
TUBE TECH meets INTEGRATED CIRCUITS

TUNG-SOL
MADE IN RUSSIA
07 01
TUNG-SOL
ELECTRON TUBE
Gold Lion
CUSTOM TUBE
Gold Lion
CUSTOM TUBE
Gold Lion
Made in Russia
Genalex
MADE IN RUSSIA
THE MASTER VALVE
Mullard
THE MASTER VALVE
Mullard
EL34
MADE IN RUSSIA
04 12

May 2015

Columns

Departments



Page 10

Nuts & Volts (ISSN 1528-9885/CDN Pub Agree #40702530) is published monthly for $26.95 per year by T & L Publications, Inc., 430 Princeland Court, Corona, CA 92879. PERIODICALS POSTAGE PAID AT CORONA, CA AND AT ADDITIONAL MAILING OFFICES. POSTMASTER: Send address changes to **Nuts & Volts, P.O. Box 15277, North Hollywood, CA 91615** or Station A, P.O. Box 54, Windsor ON N9A 6J5; cpcreturns@nutsvolts.com.

Learn more

JOIN THE EVOLUTION.

Evolve to app-based control with AIR for Wiced Smart!

AIR™ Anaren Integrated Radio for WICED by BROADCOM

If you're ready to evolve from fixed control panels populated with dials, buttons, keypads, and LCD displays to mobile-app based control of your embedded product – check out Anaren's AIR for Wiced Smart module, featuring Broadcom's Wiced Smart *Bluetooth*® chip (BCM20737). Not only does our small-footprint, SMT, and pre-certified all-in-one module save you the time, effort, and trouble of designing your own radio... It's supported by our industry-exclusive *Atmosphere* development ecosystem that lets you develop your basic embedded code and app code in one, easy-to-use development tool – for a far speedier product development cycle and time-to-market. **Follow the steps at left to join the evolution, right now!**

www.anaren.com/AIRforWiced
800-411-6596
In Europe: 44-2392-232392

Anaren®
What'll we think of next?™

Bluetooth®

Get "mobile smart" in 3 easy steps:

1 Get your ***AIR for Wiced Smart*** dev kit at your distributor of choice. (See our website for a current list.)

2 Develop your wireless link and basic app using our exclusive ***Atmosphere*** development tool.

Anaren ATMOSPHERE

3 With our AIR for Wiced Smart module on board, proceed in record time to a prototype and final, mobile-app development!

by Bryan Bergeron, Editor

DEVELOPING PERSPECTIVES

The eBay Treasure Hunt

Online sourcing from China is a real boon to electronics experimenters on a budget. Akin to flying on standby, if you're willing to wait anywhere from a week to a month for delivery, components and circuit modules can be had for less than the cost of shipping alone of similar items from domestic sources.

For example, I just finished building a heated mug for keeping my shaving brush and shaving cream warm. My initial design was based on four ceramic power resistors attached with thermal epoxy to the mug. A temperature sensor and Banana Pi (overkill, I know) running a PID (proportional, integrative, differential) algorithm and power MOSFET rounded out the simple resistor-based heater circuit. I priced out a simple thermal probe — with postage — at about $10 from a popular domestic supplier.

Thinking I might find a better deal on eBay, I searched for a similar sensor online. Not only did I find a thermal sensor, but it was attached to a complete Arduino-based PID control board with a three-digit LED temperature display and power MOSFET switch to drive the resistors. All this was for $11 — including shipping — from China. I found a dozen power ceramic resistors (also from China) for $3 including shipping. Needless to say, I ditched my original design which — by comparison — was simply cost prohibitive.

A month after making the original order for parts, I have a heated mug that works better than expected. I set the temperature to 110°F using the digital display for a guide, store my brush in the mug between latherings, and life is good. The PID controller drives the power MOSFET which cycles 12 VDC through the ceramic resistors. The thermal probe is attached to the side of the mug, providing temperature feedback. However, I can't really recommend the circuit to others because I have no way of knowing how long the circuit board will be available.

It turns out that if you're using a no-name Chinese source for your parts, you're fine — as long as you're building one-offs. However, when repeatability is an issue — such as when you or others need to create additional circuits — there's often no way to know if a vendor will have items in stock in the future, and often no clear way to find an alternate source for the same items.

For these reasons, you won't find many projects in *Nuts & Volts* that call for circuit boards sourced offshore from eBay. That's not to say that everything has to be purchased at your favorite domestic supplier — resistors are resistors, after all.

So, what's the point of buying a completed circuit online? First, there's a lot to be said for an experimenter who can integrate circuit modules. Second, who said experimentation in electronics had to involve low-level circuit designs? Sure, I wanted to work with the recently released Banana Pi and practice tuning my own PID algorithm, but my limited time and resources left no real options.

Before you embark on that new project, you might want to take a look at what's available from the overseas suppliers on eBay. Some things are worth the wait. **NV**

NUTS AND VOLTS
EVERYTHING FOR ELECTRONICS

Published Monthly By
T & L Publications, Inc.
430 Princeland Ct.
Corona, CA 92879-1300
(951) 371-8497
FAX **(951) 371-3052**
Webstore orders only **1-800-783-4624**
www.nutsvolts.com

Subscription Orders
Toll Free **1-877-525-2539**
Outside US **1-818-487-4545**
P.O. Box 15277
North Hollywood, CA 91615

FOUNDER
Jack Lemieux

PUBLISHER
Larry Lemieux
publisher@nutsvolts.com

ASSOCIATE PUBLISHER/ ADVERTISING SALES
Robin Lemieux
robin@nutsvolts.com

EDITOR
Bryan Bergeron
techedit-nutsvolts@yahoo.com

VP OF OPERATIONS
Vern Graner
vern@nutsvolts.com

CONTRIBUTING EDITORS
Fred Eady
Lou Frenzel
Ward Silver
J.W. Koebel
Roger Secura
Tim Brown
Ron Hackett
Andrew Retallack
George Steber
Dick Whipple

CIRCULATION DEPARTMENT
subscribe@nutsvolts.com

SHOW COORDINATOR
Audrey Lemieux

WEB CONTENT
Michael Kaudze
website@nutsvolts.com

WEBSTORE MARKETING
Brian Kirkpatrick
sales@nutsvolts.com

WEBSTORE MANAGER
Sean Lemieux

ADMINISTRATIVE STAFF
Debbie Stauffacher
Re Gandara

Printed in the USA on SFI & FSC stock.

READER FEEDBACK

KISS Timer is aMISS

I expect that you will probably get more comments on the KISS power article in the April 2015 issue. There are some problems; the most serious being that the instructions for using an isolated voltage source are wrong. As described, the instructions do not isolate as the 120V neutral wire is still connected to the 'isolated' power source. There is no instruction to disconnect the neutral wire. The other issue is connecting the load power to pin 11 would bypass the timer module.

The diagrams for the AC input and the LOAD output are misleading. "GND' should be the center pin and should be marked "GreenWire Ground." "Hot" should be "L1" (black) and "Neu" should be L2 (white). Also looking at the picture of the actual construction, it looks like the "Hot" from 'AC in' is connected to S4 and then to F1, rather in the way it should be per the schematic.

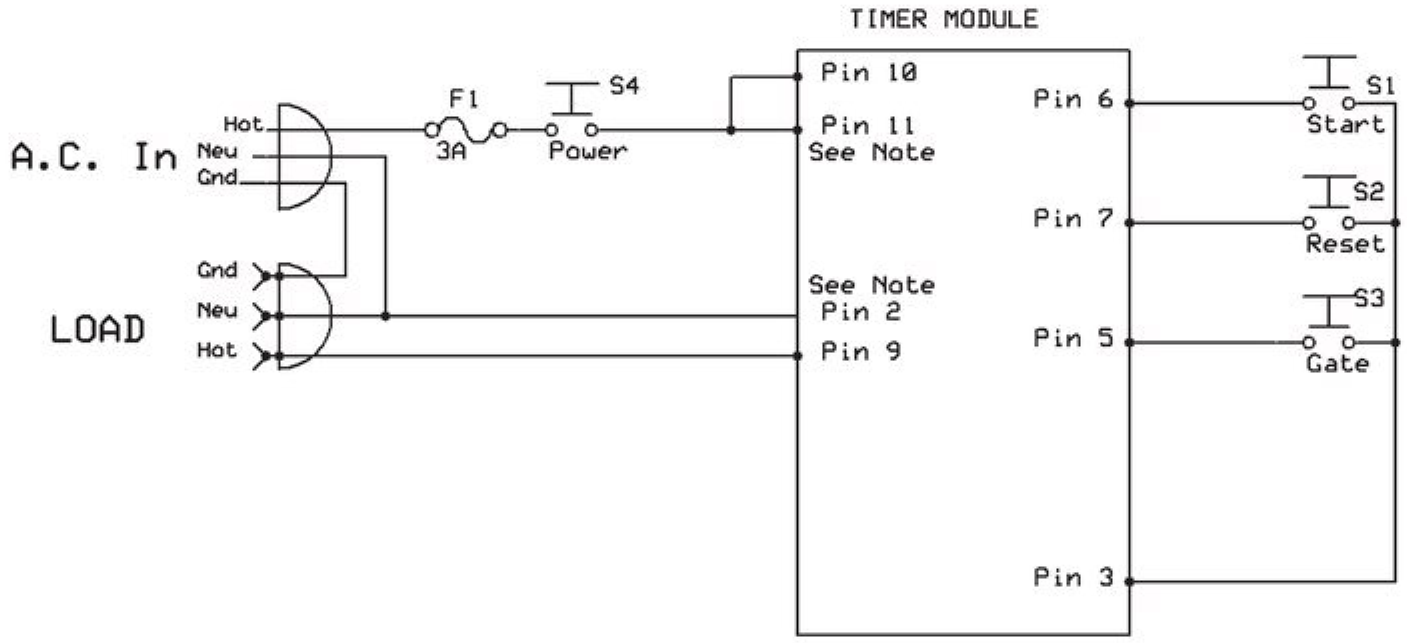
The most serious issue is leaving the neutral wire connected to the isolated power source for the timer module's relay. Battery leads should NEVER be connected to any AC power lines. You guys really should get someone to review this stuff before it is published.

Marc Fogey

Marc, I appreciate your input. I have eliminated the battery backed reference completely from the schematic (as the manufacturer's literature should have made clear) to avoid any confusion. Also, a schematic diagram should not indicate physical placement (unless it notes it), so the neutral and ground wires can be shown in any location. As you pointed out, I agree that the physical picture is somewhat misleading as the fuse should come first as it is in the schematic. I have created an updated schematic shown here to address your concerns.

Frank Muratore

Continued on page 77



NOTE:

FOR ISOLATED OUTPUT CONNECT PIN 11 TO EXTERNAL VOLTAGE

DISCONNECT PIN 10 TO 11 JUMPER

CONNECT EXTERNAL RETURN TO LOAD NEUTRAL SOCKET

AND DISCONNECT A.C. NEUTRAL WIRE TO SOCKET

A.C lead Designations:

Hot leads are Black

Neutral Leads are White

Ground Leads Are Green

Q & A

WITH TIM BROWN

In this column, Tim answers questions about all aspects of electronics, including computer hardware, software, circuits, electronic theory, troubleshooting, and anything else of interest to the hobbyist. Feel free to participate with your questions, comments, or suggestions. **Send all questions and comments to: Q&A@nutsvolts.com.**

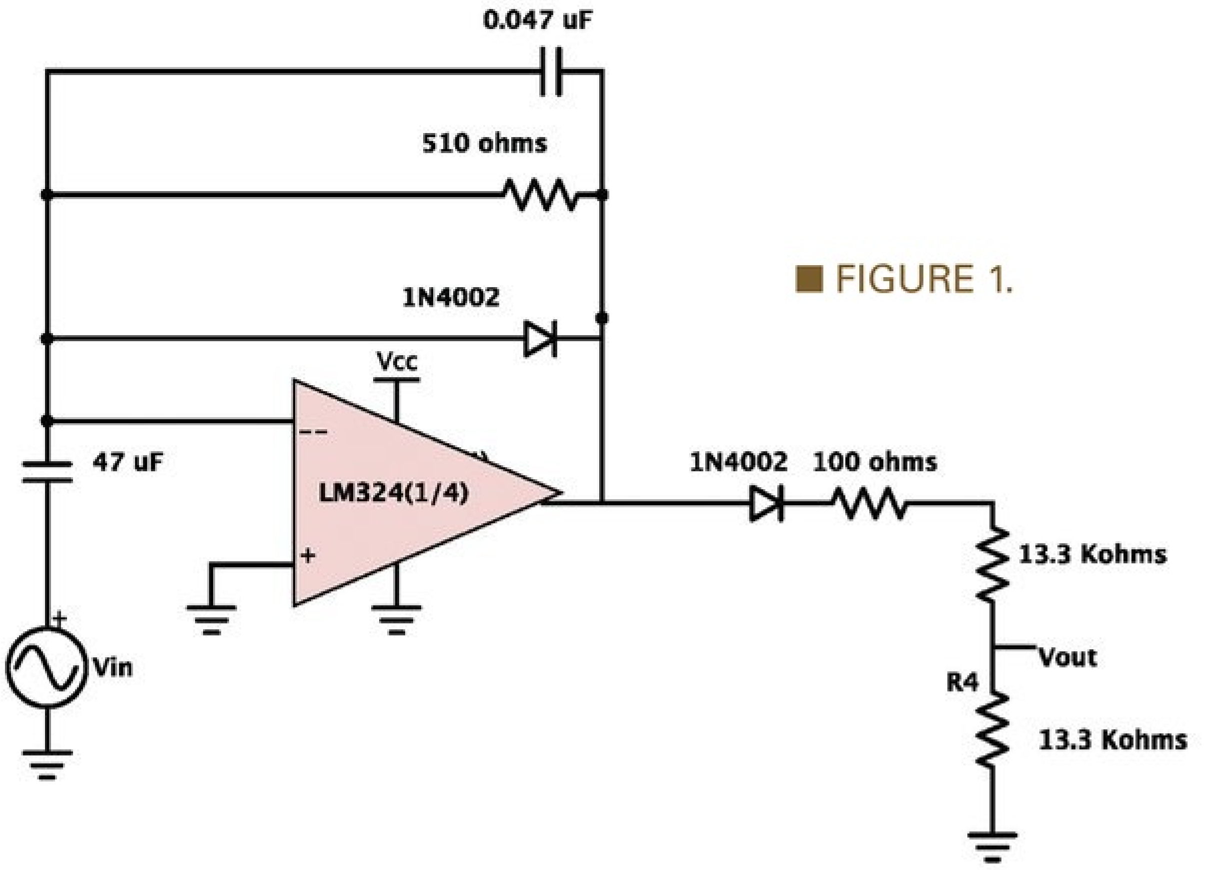
- **Op-Amp Accuracy Question**
- **Certified Electronics Technician**
- **Q&A TIPS — Batteries**

Post comments on this article at **www.nutsvolts.com/index.php?/magazine/article/may2015_QA.**

Op-Amp Accuracy Question

Q For years, I have used a variation of the circuit in **Figure 1**. The output was reliably zero, plus or minus 0.1 millivolts or so, with no input. Lately, almost all LM324s I buy — regardless of price — are likely to be plus or minus one millivolt, and I discard most of them as useless. Can anyone tell me where to get a quad op-amp that has reliable low offset?

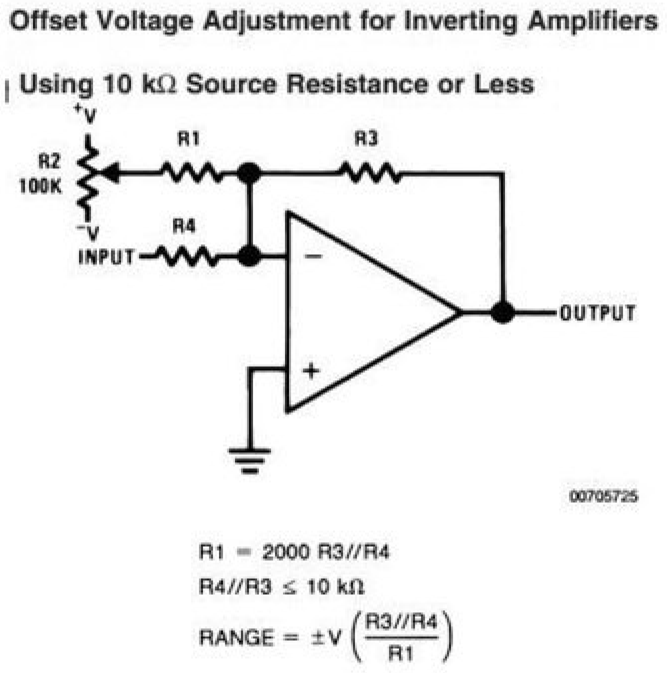
— Milton Lilie
via email



■ FIGURE 1.

A You are talking about what is known as output offset voltage which is the value of voltage at the output terminal of an operational amplifier (op-amp), with zero voltage across the input terminals (shorted input terminals). The op-amp is a high gain/DC-coupled differential amplifier with a large bandwidth, high input impedance (doesn't heavily load the previous circuit), and low output impedance (dissipates less energy in output stages) which — in modern times — is packaged in a single small Integrated Circuit (IC) device (you can also make op-amps with vacuum tubes).

Op-amps act as instrument amplifiers (increase level of millivolt sensor outputs), comparators, summing circuits, etc. Op-amps are constructed of transistors, diodes, resistors, and capacitors which are "formed" on silicon substrates (kind of like micro versions of printed circuits). Like discrete components, these epitaxial components have tolerances which means the electrical characteristics of different devices on the same chip vary.



■ FIGURE 2.

Adapted from Stack Exchange (http://electronics.stackexchange.com/questions/34071/how-do-i-correct-offset-voltage-of-opamps-which-have-no-explicit-offset-null-pin}

The op-amp input is a differential (difference) amplifier made up of pairs of transistors which roughly oppose each other. Herein lies the output offset problem. If the transistor's characteristics (mostly the transistor gain, *hfe*) exactly match, when the input to the + (non-inverting) and - (non-inverting) terminals is equal, the output voltage will be zero.

In the real world, transistor characteristics match exactly on very rare occasions (maybe influenced by the lunar-planetary alignment ???).

In order to correct the output offset voltage, we introduce an input offset voltage to force the op-amp's output to zero when the + and - terminal voltages are equal. I am assuming that your circuit uses the LM-324 op-amp IC as an inverting amplifier, so **Figure 2** would work as a offset voltage compensation circuit (for non-inverting amps, just inject the compensation signal into the + terminal). I cannot think of a way to compensate for an op-amp without compensation terminals, unless you use a bipolar power supply since you must be able to add either positive or negative voltage into the op-amp.

The TI LM-324 datasheet says the offset voltage is seven millivolts, so your devices are all within specification. For quad op-amps advertised as low offset voltage devices, the Analog Devices' ADA 4077-4 has a offset voltage of 10 microvolts (0.01 millivolts), so this may meet your requirements.

Unfortunately, the price is two to three times the cost of the LM-324s ($1.50 versus $0.50 per device is not too bad if you need this precision and do not want to use a compensation circuit).

Certified Electronics Technician

Q I am trying to get into the electronics field, and have seen there is a certification for a Certified Electronics Technician. I'm wondering if it is useful to obtain. and where to start.

— Joshua J. Apodaca
via email

A A good question for establishing your credentials as an electronics technician. The International Society of Certified Electronics Technicians (ISCET at the website **www.iscet.org/certification**) is a good place to start. ISCET has associate, non-journeyman, and journeyman levels of certification, with endorsements at the journeyman level in most of the electronics specialties such as industrial, communications, computer, medical, and radar. Testing fees range from $45 to $110. ISCET uses a network of Authorized Independent Certification Administrators who are located in various parts of the country and vary with the tests you need to take, so check out the website to find the nearest location. The ISCET store offers study materials.

To become an FCC Licensed Electronics Technician, check out long-time *NV* advertiser, Command Productions at **www.licensetraining.com**. They offer several excellent home study courses.

If you are interested in computer/information technology technician certifications, CompTIA (**www.comptia.org**) has certifications for initial IT entry level A+, Network+, Security+ (I have all three), Server+, etc., or the Microsoft Certified Solutions Expert (MSCE; **www.microsoft.com/learning/en-us/mcse-certification.aspx**).

Another *NV* advertiser, Cleveland Institute of Electronics (**www.cie.wc.edu**) offers courses for computer and electronics technicians, as well. **NV**

Q&A TIPS – Batteries

Here are a few tips I have discovered over the years when dealing with non-rechargeable alkaline batteries:

1. Before discarding all of the batteries from non-working electronic devices, check each one for proper voltage (alkalines should be 1.5 volts when fresh, but can usually operate devices down to a 1.3 volt end point). I have found that sometimes only one of the batteries reads low and the others are good.

2. For a non-functioning device, check the polarity of all the batteries. Years ago, I was given a radio that would not work with the batteries, but worked okay on AC power. I checked the four batteries and found all read 1.5 volts, but one of the batteries had reverse polarity. A couple of months ago, I checked the batteries on a satellite dish remote and found one battery with reverse polarity, so I replaced the battery and tried the unit. The remote's LED illuminated a couple of times, then quit again. Once again, a battery had reverse polarity so I replaced the remote.

3. Don't just throw batteries in the trash. Some jurisdictions have environmental regulations and procedures to properly recycle batteries. Batteries are loaded with nasty hazardous chemicals that we don't need in landfills.

4. I have found that keeping alkaline batteries in the freezer will extend their shelf life. I have read manufacturer's blurbs about not doing this because it causes the battery's power to decrease, but if the battery is brought to room temperature before the device is used, the batteries will have normal power (don't put the batteries in a device until they have stopped "sweating" to prevent damage to the device).

Batteries contain highly corrosive materials, so reducing their temperature reduces the rate of deterioration of the metals within the battery (which will eventually render the battery useless). I have kept batteries in the freezer for years, and when "thawed" they worked normally. Don't freeze lead-acid batteries or they may be damaged.

5. When storing or transporting nine volt batteries, avoid shorting the anode and cathode. I once inadvertently invented a pocket warmer when a battery's electrodes were shorted by a coin in my pocket. Fortunately, I felt the problem before it became a disaster.

6. With nine volt batteries in devices that do not have the polarized connectors, make sure the battery polarity is correct. Once, as a sound tech, someone installed a fresh nine volt battery but the lavalier mic would not work and the mobile unit was getting very hot over the battery compartment. In this case, we had "fried" a $100+ wireless unit.

■ BY RON HACKETT
www.jrhackett.net

PICAXE-PC Serial Communication: Part 1

In the previous installment of the Primer, we experimented with a hardware solution to inverting the data lines on a Prolific PL2303HX USB-to-serial cable, so that it can be used to power and program our PICAXE breadboard circuits. This month, we're going to continue our experiments with the Prolific cable, but this time we're going to focus on sending data back and forth between a PICAXE processor and a PC.

As you probably know, the PICAXE serial programming protocol requires inverted (negative) serial communications, which is why we had to invert the two PL2303HX data lines in the previous column. However, we won't be using the PL2303HX cable with the PICAXE programming pins this time, so we aren't limited to inverted serial data.

In PICAXE BASIC, the *serin* and *serout* commands can be utilized on any PICAXE I/O pin, and both commands can be configured to work with either positive or negative serial communications. We'll simply use true (positive) serial communications for our experiments, so the PL2303HX cable will provide a very inexpensive and straightforward means of making the connection between a PICAXE project and a PC.

This month, we'll conduct four simple software experiments that explore the process of establishing a PICAXE-PC serial communications link via the PL2303HX cable.

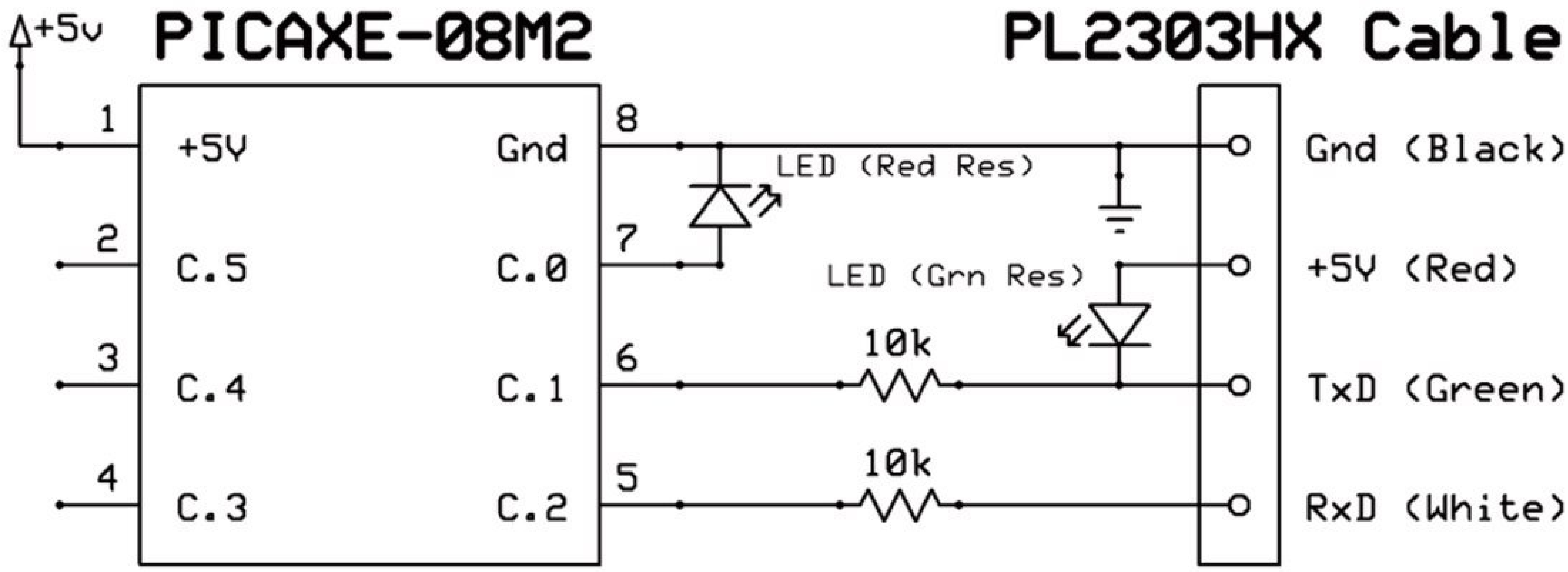
Figure 1 presents the schematic of the circuit that we'll be using for all four experiments. As you can see, it doesn't include the standard PICAXE programming circuit; you can just use whatever programming adapter you prefer.

The two 10K resistors in **Figure 1** serve as current-limiting resistors to protect pins C.1 and C.2 on the 08M2, as well as the TxD pin on the Prolific cable. As you know, pins C.1 and C.2 can each be configured either as an input or an output. Of course, that flexibility can create problems if the PL2303HX pins are connected incorrectly and/or if either of the 08M2 pins is configured incorrectly.

For example, if we didn't include a current-limiting resistor on pin C.1 and we accidentally configured C.1 as an output while it was connected to the TxD pin on the Prolific cable, either or both of the pins could be damaged or destroyed. The current-limiting resistor prevents any possibility of accidental hardware damage.

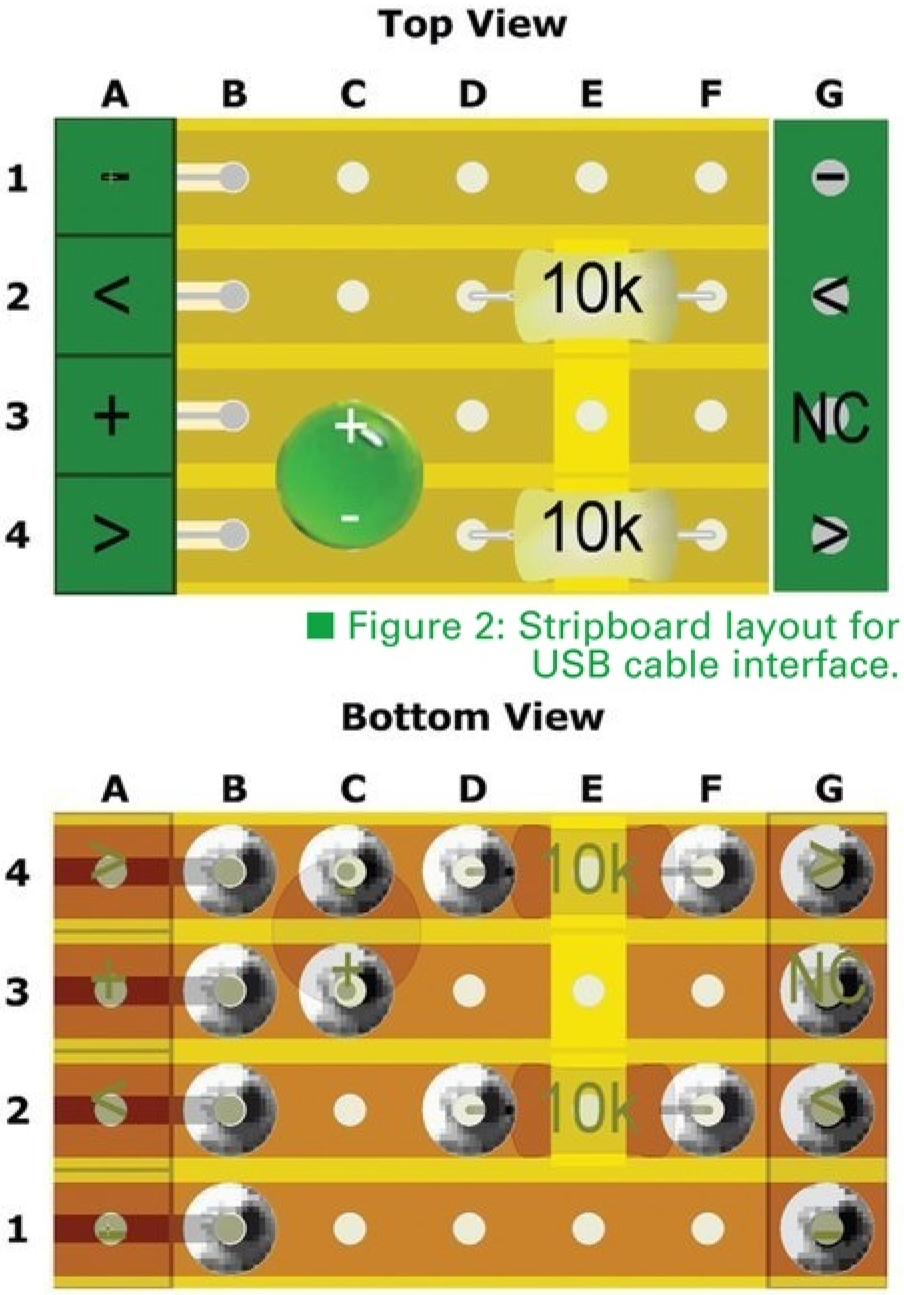
The red resistorized LED connected between pin C.0 and Gnd simply serves as a "debugging" LED. In all our experiments this month,

■ Figure 1: Schematic for USB cable interface.



■ Figure 3: Parts List for USB cable interface.

Stripboard, 4 Traces with 7 Holes Each
Resistor, 10k, 1/6W (2 pcs.)
LED, 3mm, Green, Resistorized
Header, Male, Straight, Reverse-Mounting, 4-pin
Header, Male, Right-Angle, 4-pin



■ Figure 2: Stripboard layout for USB cable interface.

we'll blink the LED so that we can see whether the program is running correctly.

The green resistorized LED that's connected between the +5V and TxD pins on the Prolific cable will flicker briefly any time serial data is being transmitted by the PC. (Since we're using positive serial transmission, the TxD line idles high and pulses low when data is being transmitted.) Also, note that the +5V pin on the Prolific cable isn't connected to the breadboard circuit at all; our breadboard will include its own power supply, so we certainly don't want to have two different +5V power connections in the same circuit.

The circuit presented in **Figure 1** can be easily implemented on a breadboard, but we'll be using it again in the next installment of the Primer, so I decided to construct a simple stripboard version of it. The stripboard layout is shown in **Figure 2**, and a larger version is available for downloading at the article link. The complete Parts List for the circuit is shown in **Figure 3**.

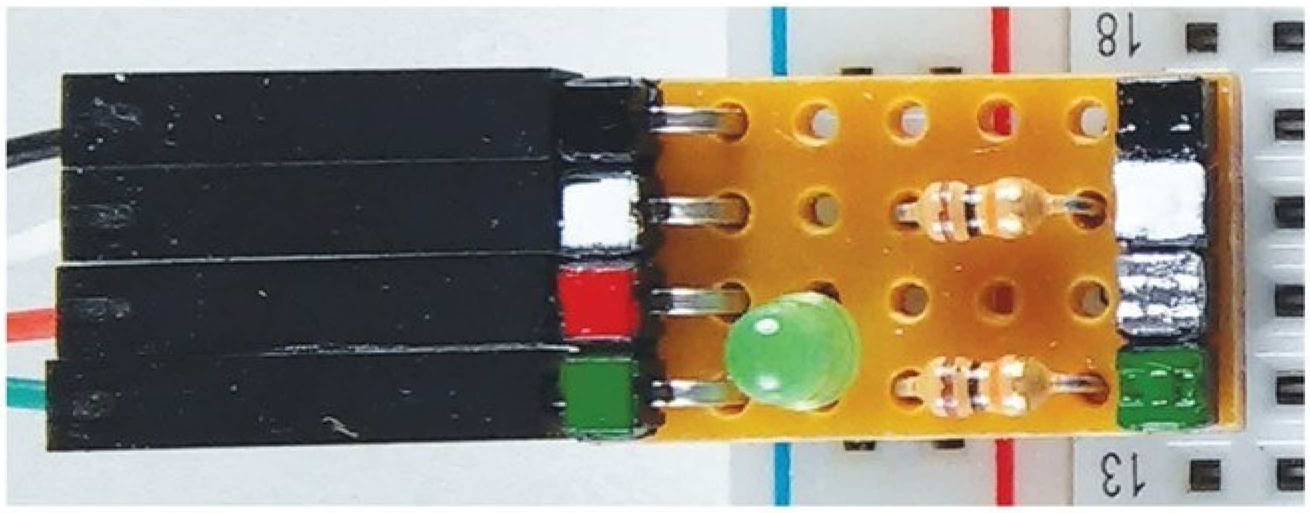
Our little stripboard circuit is so simple and we've been doing this for so long, that I think we can dispense with the usual list of assembly instructions. However, I should point out that the four-pin straight header in column G needs to be the "reverse-mounting" type so it can be inserted from the top of the stripboard and soldered on the bottom.

If you prefer the safety and convenience of a stripboard circuit, just build it! If not, you can take the more traditional "rat's nest" approach, and assemble the circuit directly on your breadboard.

If you do decide to construct the stripboard circuit, you may want to add four little spots of paint to help you remember which wire on the Prolific cable gets inserted onto which pin. To see what I mean, refer to the close-up photo of my completed stripboard circuit that's in **Figure 4**. (I used silver paint to label the pin that's not connected to anything.)

Whichever approach you choose, my breadboard setup for all this month's experiments is presented in **Figure 5**. As you can see, I'm using our standard +5V breadboard power supply and the Prog-03 programming adapter for the AXE027 USB-to-serial cable. Of course, you can use a variety of other hardware to accomplish those two tasks.

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/may2015_PICAXEPrimer**.



■ Figure 4: Completed stripboard circuit.

At this point, I should justify my choice of an 08M2 processor for this month's experiments. As you can see in **Figure 5**, the PICAXE programming interface and the Prolific cable interface require connections to I/O pins C.0, C.1, C.2, and C.5, which means that C.3 (input only) and C.4 (I/O) are the only two 08M2 pins that remain available for other program functions.

However, our only goal this month is to compare the use of the *serin* command verses the *hserin* command for serial communications with a PC, and the 08M2 processor is more than adequate for that purpose. When you're ready to implement a serial communication project that requires more than two free I/O pins, the stripboard can also be easily connected to any current PICAXE processor.

In the next Primer, we'll use the same stripboard circuit with a 20M2 processor; the connections are even simpler in that case. In fact, I actually designed the stripboard circuit for use with a 20M2 or 20X2 processor.

Before we move on to implementing our first experiment, there's one final point I need to clarify: We're not going to use the *hserout* command in this month's experiments. On both the 08M2 and 14M2 processors, that command is only available on the serial out pin (C.0 on the 08M2, and B.0 on the 14M2).

Consequently, if we were to use the *hserout* command, every time we downloaded a new or updated program to the processor, we would also be sending "garbage" characters out to the PC. In order to avoid that problem, we'll be using the *serout* command with pin C.2 on the 08M2 (or pin B.2 on the 14M2). In the upcoming July column when we experiment with 20M2-based PICAXE-PC serial communication, we will definitely use the *hserout* command.

In each of the experiments this month, we're simply going to send small amounts of data back and forth between our 08M2 breadboard circuit and a PC. The procedures involved will be essentially the same whether you're using a Macintosh, Linux, or Windows computer.

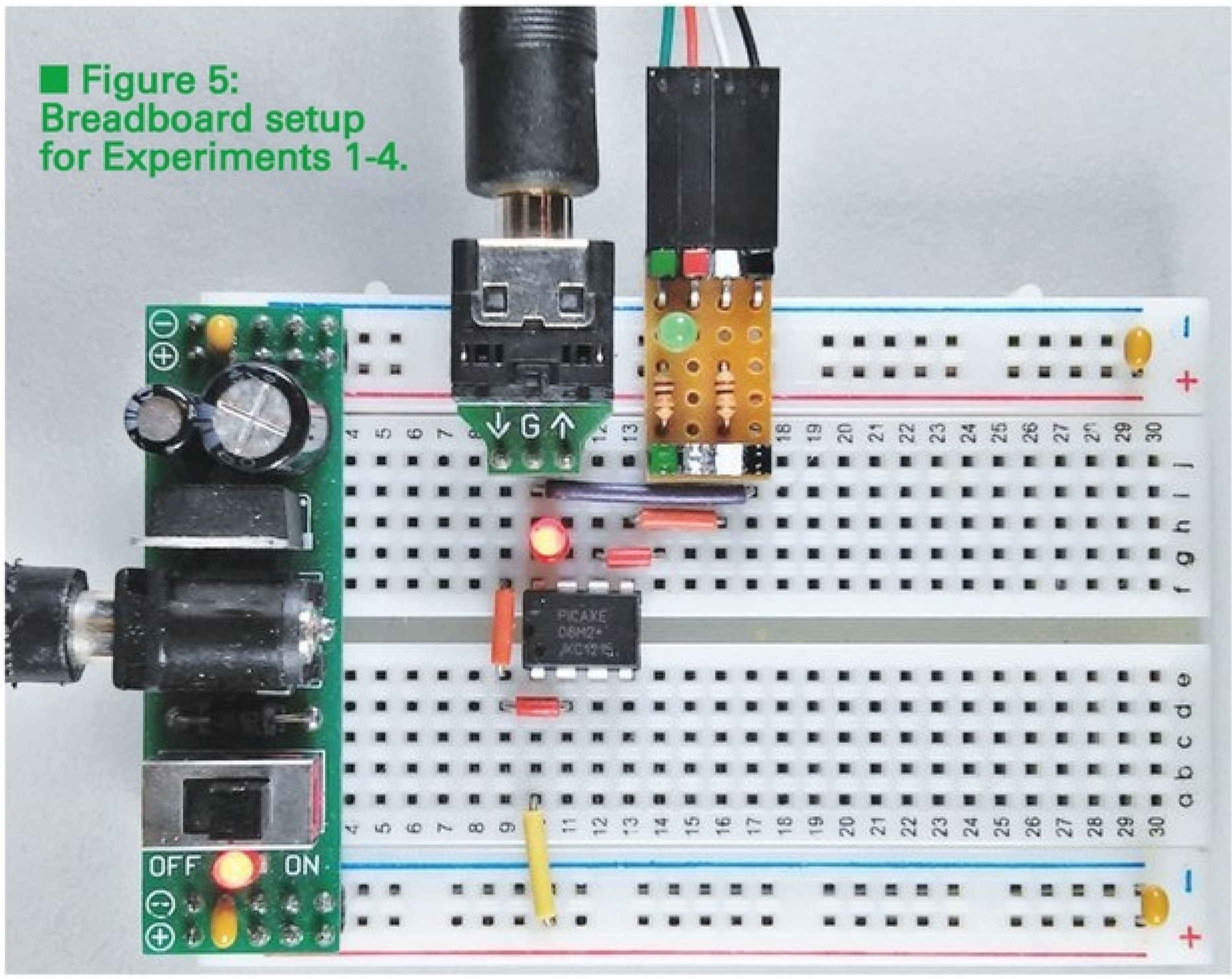
On the PC side of the communications link, you will need to have a serial terminal program installed. You could simply use the terminal in the PICAXE Editor for this purpose, but a separate terminal application will provide a couple of significant advantages — especially when we tackle more involved experiments in future columns.

My suggestion would be to install a separate serial terminal application on your PC (if you don't already have one), and use it right from the beginning. I chose to use the CoolTerm application because it's highly rated, it's available for all three major operating systems, and it's free.

For the remainder of this article whenever I mention a detail about using the terminal program, I'm referring to CoolTerm. If you also decide to use CoolTerm, you can easily find a download site with a quick Google search. Once you have it installed, take some time to read through the documentation that's available by clicking on the "Help" icon in the toolbar. If you're using a different terminal program, you may need to read its specific documentation.

In all four experiments this month, we'll need to have two different programs running at the same time: the PICAXE Editor, and the terminal of your choice. If you're fortunate enough to have more than one monitor attached to your PC, the easiest approach is to have each program running on its own monitor. If not, just size the two program windows so you can see both of them at the same time, rather than having to switch back and forth between the two programs.

Finally, the four programs that we'll be using this month are



■ Figure 5: Breadboard setup for Experiments 1-4.

available for downloading at the article link. You may want to take a break at this point, and download them before moving on to our first experiment.

Experiment 1: Using the *Serout* Command to Send Data to a PC

The main purpose of our first experiment is to make sure that we've correctly configured our communication link, and to become comfortable with the process of communicating with a PC. Therefore, this time we're only transmitting serial data in one direction (PICAXE to PC). In subsequent experiments, we'll move on to explore two-way communication.

■ Figure 6: *SeroutToPC.bas* program listing.

```
01    ' === SeroutToPC.bas =============================
02    ' Simple program to verify communication link
03
04    ' === Directives ===
05      #com 4                  'specify serial port
06      #picaxe 08M2            'specify processor
07      #terminal off           'disable terminal window
08      #no_data                'save time downloading
09
10    ' === Constants ====
11      symbol   LED = C.0      'debugging LED on C.0
12      symbol RxPin = C.1      'connect to TxD on USB cable
13      symbol TxPin = C.2      'connect to RxD on USB cable
14
15    ' === Begin Main ==================================
16    setfreq m8                        'so we can use 9600 baud
17
18    do
19      high LED                                  'LED on
20      serout TxPin,T9600_8,("On... ")           'send LED state
21      wait 2                                    'wait 1 second
22      low LED                                   'LED off
23      serout TxPin,T9600_8,("Off",cr)           'send LED state
24      wait 2                                    'wait 1 second
25    loop
```

When you've completed your breadboard setup for the experiment, the next step is to determine which serial port the Prolific cable is using. To do so, just run your terminal program and look at the list of ports the program sees. (In CoolTerm, that list is available under the "Options" >> "Serial Port" menu.)

Next, plug in the Prolific cable, re-scan the serial ports, and look at the list again; the newly added port is the one that's connected to the Prolific cable.

When you've selected the correct serial port, the next step is to configure the options for serial communication. CoolTerm only supports positive serial communications, but that's exactly what we want anyway. I'm using 9600 baud (8-N-1) with no flow control, but you can certainly make that faster if you want. Just be sure the PICAXE configuration matches what you select in the terminal.

CoolTerm (and I assume most other terminal programs) has two "Terminal Modes" of data transmission: Raw Mode and Line Mode. In Raw Mode, a data byte is transmitted from the PC as soon as any key is pressed on the keyboard; in Line Mode, nothing is transmitted until the Enter key is pressed, and then the entire series of key presses is transmitted.

For all our experiments this month, we'll use the Raw Mode of data transmission. In CoolTerm, you can change the Terminal Mode under the "Options" >> "Terminal" menu. However, Raw Mode is the default, so you can just leave it alone.

There are several other parameter settings that are available under the CoolTerm Options menu but there's nothing else we need to configure, so let's move on to the PICAXE software for our first experiment (*SeroutToPC.bas*) which is shown in **Figure 6**. The program is very simple, but there are a few points that I should mention. Each of the following numbers refers to the corresponding line number at the left edge of **Figure 6**:

07: Here, we disable the PICAXE terminal because we're using CoolTerm.

12: On the 08M2 and 14M2 processors, pin C.1 is the *hserin* pin, but that pin can also be used with the more traditional *serin* command. If you read through the program listing, you'll see that we aren't using pin C.1 this time, but I decided to include this constant declaration anyway since we'll be using it in the remaining experiments this month.

16: We're going to transmit serial data at 9600 baud, which is not available at the 08M2 default system clock rate of 4 MHz. Here, we specify 8 MHz so that we can use 9600 baud. (See the PICAXE documentation for the *serout* command for details.)

21: Don't forget! When we double the system clock rate, we also halve all *delays* produced by the *wait* and *pause* commands. For example, the *wait 2* command results in a one second pause.

When you've connected the Prolific cable between your breadboard setup and PC, and you've configured your terminal program to receive data from the 08M2 processor, download the *SeroutToPC.bas* program to the 08M2. You should see the "On" and "Off" transmissions appearing in your terminal window, in time with the blinking of the LED on pin C.0 of the 08M2;. If not, you will need to troubleshoot your setup.

Experiment 2: Using the *Serin* Command to Receive Data from a PC

In this experiment, we're going to attempt to implement two-way communication between the 08M2 and the PC by using the PICAXE *serin* command. In the process, we'll encounter a major problem with the use of this command.

The breadboard setup and the PICAXE-PC connection remain the same in this experiment (and all the remaining experiments this month), so all we need to do is download a different program (*SerinBlockingFromPC.bas*) to the 08M2 processor.

If you read through the program listing for *SerinBlockingFromPC.bas*, you will see that it's essentially the same as the *SeroutToPC.bas* program we just used, with the addition of the following two program lines:

```
serin  RxPin,T9600_8,cmd
serout TxPin,T9600_8,("cmd =
  ",cmd,cr,lf)
```

The purpose of the above *serin* statement is to receive a single key press from the terminal program, and to store its value in the *cmd* variable. The *serout* statement simply sends the same value back to the PC to confirm that the "command" was received. However, there's a major problem with our first attempt at two-way PICAXE-PC communication which I bet you've already anticipated!

Before you download the *SerinBlockingFromPC.bas* program to your breadboard setup, there's an important point to mention. As you remember, CoolTerm is configured to transmit data in Raw Mode by default, which means that a data byte is transmitted from the PC as soon as any key is pressed on the keyboard. However, in order for this to work, CoolTerm has to be the currently active program. In other words, after you download the software program to your breadboard setup, you have to click in the CoolTerm window to make it the active application on your PC. If you forget to do that, CoolTerm will not be transmitting any data at all!

So, what's wrong with our program (he asks rhetorically)? As you probably remember, *serin* is a "blocking" command by default; the program just waits at the *serin* command in this experiment, until a character is received. In other words, nothing at all happens until you press a key on the keyboard.

At that point, the character is echoed back to the terminal, the LED on pin C.0 blinks (only once), and the program again waits at the *serin* command for as long as it takes to receive another character from the PC. If the 08M2 had more important tasks than simply blinking an LED, those tasks wouldn't be accomplished either.

Obviously, this is a completely unacceptable situation for any "real" data processing project. At first glance, it might seem the new *timeout* option for the M2-class version of the *serin* command will fix this problem. However, the use of *timeout* has its own problems, as we're about to see in our next experiment.

Experiment 3: Using *Timeout* with the *Serin* Command

The PICAXE *serin* command now includes several optional syntaxes that are available for all M2, X1, and X2 processors. (For complete details, see the documentation for the *serin* command in Section 2 of the PICAXE manual.) The syntax for the version of the *serin* command that we will use in this experiment is as follows:

```
serin [timeout], pin, baudmode,
  variable
```

As we just saw in Experiment 2, the default syntax for the *serin* command is blocking: program execution stops at the *serin* instruction, and waits (possibly forever) until serial data is received. However, when we include the optional *timeout* parameter, if no serial data is received within the time specified by *timeout* (in milliseconds), program execution automatically resumes.

In order to clarify how this works, let's examine the following subroutine:

```
getCmd:
  cmd = 0
  serin [2000],RxPin,T9600_8,cmd
  if cmd > 0 then
    serout TxPin,T9600_8,("cmd =
    ",cmd,cr,lf)
  endif
```

In the first line of the above subroutine, we initialize the *cmd* variable to 0, so that we can use that value as a "no command received" flag. Next (when the *serin* statement is executed), if no command is received within one second (1,000 ms, because we're running at 8 MHz), the program moves on to the *if/then* statement. Since the value of the *cmd* variable is still 0, the *serout* statement is skipped.

On the other hand, if the *serin* statement does receive a command byte within the one second timeout period, the program immediately continues execution at the *if/then* statement. Since the value of the *cmd* variable has been updated from 0 to some positive value, the *serout* statement is executed and the value of the command byte is sent to the PC.

With that explanation in mind, download *SerinTimeoutFromPC.bas* for this experiment to your breadboard setup, and make sure that your terminal program is also running. As long as you don't enter a command character from the PC keyboard, the program behaves properly: The LED blinks repetitively, and the 08M2 transmits the real time

LED state to the terminal.

However, the problem becomes apparent when you begin to send command characters to the 08M2. (Don't forget that you need to first click inside the CoolTerm window for the characters to be sent.) Spend some time sending command characters to the 08M2; some will be transmitted to the 08M2 (and echoed back to the terminal), and others will be lost. See if you can diagnose the cause of the problem before reading further. Here's what you should have observed:

- Whenever a key press occurs when "On" is displayed in the last line of the serial terminal, the command is not received or echoed by the 08M2.
- Whenever a key press occurs when "On ... Off" is displayed in the last line of the serial terminal, the command is received and echoed by the 08M2, and the LED is immediately turned on again.

Examine the code in the main *do/loop* of the *SerinTimeout FromPC.bas* program, and these results should make sense. The *serout* statement that echoes "Off" is the last one in the main *do/loop*, so the program immediately loops again, initializes the *cmd* variable, and the *serin* statement waits for serial input for one second.

If you press a key during the timeout period, it's captured by the *serin* statement and echoed back out. On the other hand, if the serial input occurs during the one second that the LED is on, the *serin* statement is not active so the data is missed.

At this point, we can certainly conclude that the programs we tested in Experiments 2 and 3 are totally inadequate for the purpose of receiving serial input from our PC terminal. There are several techniques we could use to correct the deficiencies. In fact, I think we've already covered one of them (software interrupts) in an earlier Primer column, but I haven't been able to find the specific reference.

In any case, the effective use of software interrupts with serial inputs can be a complicated topic. However, all M2-class processors also include a simpler and much more powerful capability: hardware serial input.

Experimenting with Hardware Serial Input

Hardware serial input is not a new capability for PICAXE processors. In fact, we discussed this topic way back in the February 2009 Primer. However, at that time, the M2-class processors hadn't yet been released. Fortunately for us, when they did become available, the M2 processors were also able to implement hardware serial input.

The real power of hardware serial input is that — after the initial setup procedure — the serial input can occur completely in the background while a program is executing other code. In other words, we're completely freed from any of the timing problems we just discussed.

The hardware serial input capability of M2 processors is not nearly as powerful as that of the X1 and X2 processors. For example, the 28X1, 40X1, and 20X2 processors are able to receive up to 128 serial bytes in the background at one time, and the 28X2 and 40X2 processors can receive as many as 1,024 serial bytes in the background.

However, all M2 processors are limited to just two bytes of background serial data. That may seem like a huge limitation, but one or two bytes of serial input data are more than ample for what I have in mind.

For example, imagine an M2-based data collection system that's serially linked to a PC that's running a "master" program. How are we going to implement a master program on a PC, you ask? Think Python: We've already covered the basics in our PICAXE-Pi explorations, and Python is freely available for all three major PC operating systems. (I digress ... we'll discuss PC-based Python "master" programs in future columns.)

For now, let's simply focus on this question: What does a PC-based master program need to communicate to a PICAXE-based data collection system?

The answer, of course, is that the master program needs to send commands to the slave data collection system. Even if we limit the master program to transmissions of one data byte at a time, the master can issue 255 different commands. (Don't forget, we need to reserve the value 0 as a "no command" flag.) I don't know about you, but I have no plans to implement a system that requires more than 255 different commands!

With that in mind, let's return to the topic of M2-based hardware serial input. In order to configure hardware serial input on an M2 processor, all we need to do is issue an *hsersetup* command at the beginning of our program. The complete syntax for this command is as follows:

```
hsersetup baud_setup, mode
```

The *baud_setup* parameter specifies the baud rate, which ranges from a minimum of 300 baud to a maximum of 115200 baud. (The latter figure is not a typo; the *hserin* and *hserout* commands can be as much as six times faster than *serin* and *serout*!)

For a complete list of valid entries for the *baud_setup* parameter, see the documentation for the *hsersetup* command in Section 2 of the PICAXE manual. However, for the next experiment we'll just use B9600_8, which means that the baud rate will be 9600 and the 08M2 system clock is set to 8 MHz (by the *setfreq m8* statement).

The mode parameter of the *hsersetup* command requires a five-bit binary number, whose bits specify various special functions of the hardware serial port. In the next experiment, we're going to use %01000 for the mode parameter. The following information explains the chosen value for each bit:

bit0: Receive background serial data to the scratchpad [1 = use scratchpad]. We must use 0 because M2 processors don't contain a scratchpad.

bit1: Invert serial output data [1 = negative]. We'll use 0 [true] for output because M2 processors must use "true" for input. (See *bit2* next.)

bit2: Invert serial input data [1 = negative]. We must use 0 [true] for input because M2 processors are not able to receive inverted serial data.

bit3: Disable *hserout* [1 = *hserout* pin is normal I/O]. We'll use 1 [disable] because we're not going to use the *hserout* pin for that purpose.

bit4: Disable *hserin* [1 = *hserin* pin is normal I/O]. We'll use 0 [don't disable] because we're going to use the *hserin* pin for that purpose.

To summarize all the above information, the *hsersetup* statement that we will use in the next experiment is *hsersetup B9600_8, %01000*.

Understanding the M2-Class *Hserin* Command

Now that we've specified our *hsersetup* statement, we can move on to discussing the M2-class *hserin* command. The syntax is mercifully simple: It's just *hserin myVar*, where *myVar* is the variable that receives the incoming serial data byte. However, the question of how the incoming background-received data byte actually gets transferred to the variable is a little more complicated.

As we discussed previously, M2 processors do not have a scratchpad storage area; instead, they use an internal two-byte first-in first-out (FIFO) storage buffer. A maximum of two bytes can be received in the background at any time — not just when the *hserin* command is executed. If more than two bytes are received before they are processed, the extra ones are lost.

Whenever an *hserin* command is executed, the first byte in the buffer is transferred to the *myVar* variable. If there is a second byte in the buffer, it's moved up to the first position. Finally, if the buffer is empty, the value of *myVar* is not changed.

Experiment 4: Using *Hserin* to Receive Serial Data in the Background

The program for this experiment (*HserinFromPC.bas*) includes the following subroutine:

```
getCmd:
```

```
  cmd = 0
  hserin cmd
  if cmd > 0 then
    serout TxPin,T9600_8,("cmd =
    ",cmd,cr,lf)
  endif
  return
```

Except for the *hserin* command, this subroutine is identical to the one we just tested in Experiment 3. However, the *hserin* command in non-blocking, so this time the program will function correctly. Download the program to the 08M2, and make sure that your terminal program is also running. (Again, don't forget to click in the CoolTerm window to make it the active application on your PC.)

As in our earlier experiments this month, the real time "on and off" data changes in unison with the blinking of the LED, so you can watch the terminal screen as you carry out the following four tests:

1) Every time you see "On" appear in the terminal (and before "Off" shows up), tap one key on your PC; you should see that every key press is echoed and that the timing of the blinking isn't perceptibly altered.

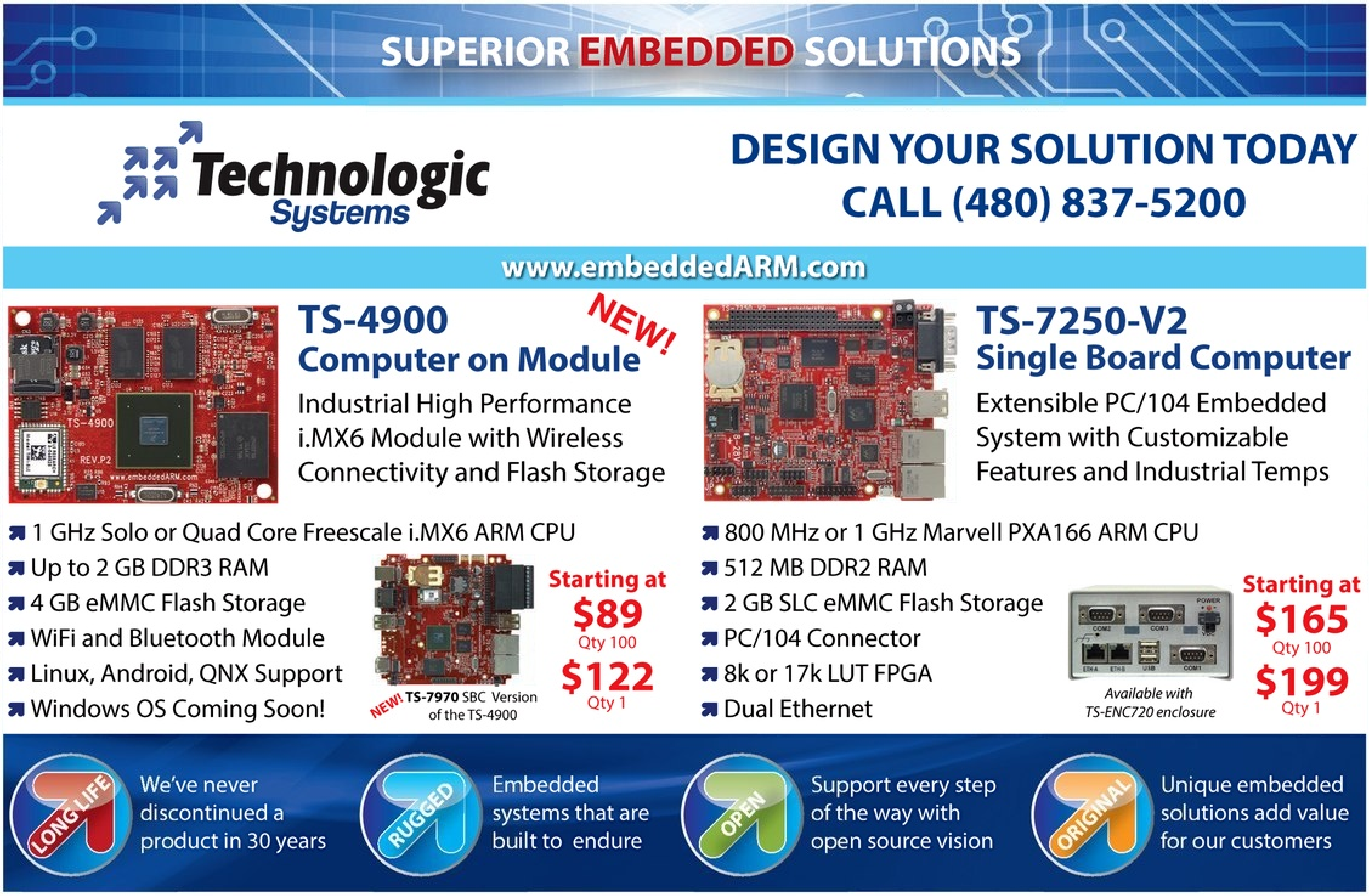
2) When you see "On" appear in the terminal (and before "Off" shows up), tap two different keys and wait to see what is echoed; you will see that both key presses are echoed, but the second echo shows up one blink later. That happens because a full blink has to occur before the *getCmd* subroutine is called again.

3) Modify the *HserinFromPC.bas* program by adding a second *gosub getCmd* statement immediately after the first one in the main *do/loop*. Download the program again, and repeat test #2. This time, you will see both characters echoed on the same line.

4) When you see "On" appear in the terminal (and before "Off" shows up), tap three different keys and wait to see what is echoed. You will see that the first two key presses are echoed, but the third key press has been lost. That happens because the M2-class FIFO buffer is only two bytes deep.

The point of these four tests is to demonstrate that as long as we (or a "master" program on a PC) do not send more than two serial bytes within a short period of time, the *hserin* command is always able to capture the serial data in the background while the M2 program is busy doing other tasks (hopefully, more important tasks than blinking an LED!).

Once again, we're out of space this month, so we'll continue our exploration of hardware serial communication next time. See you then ... **NV**



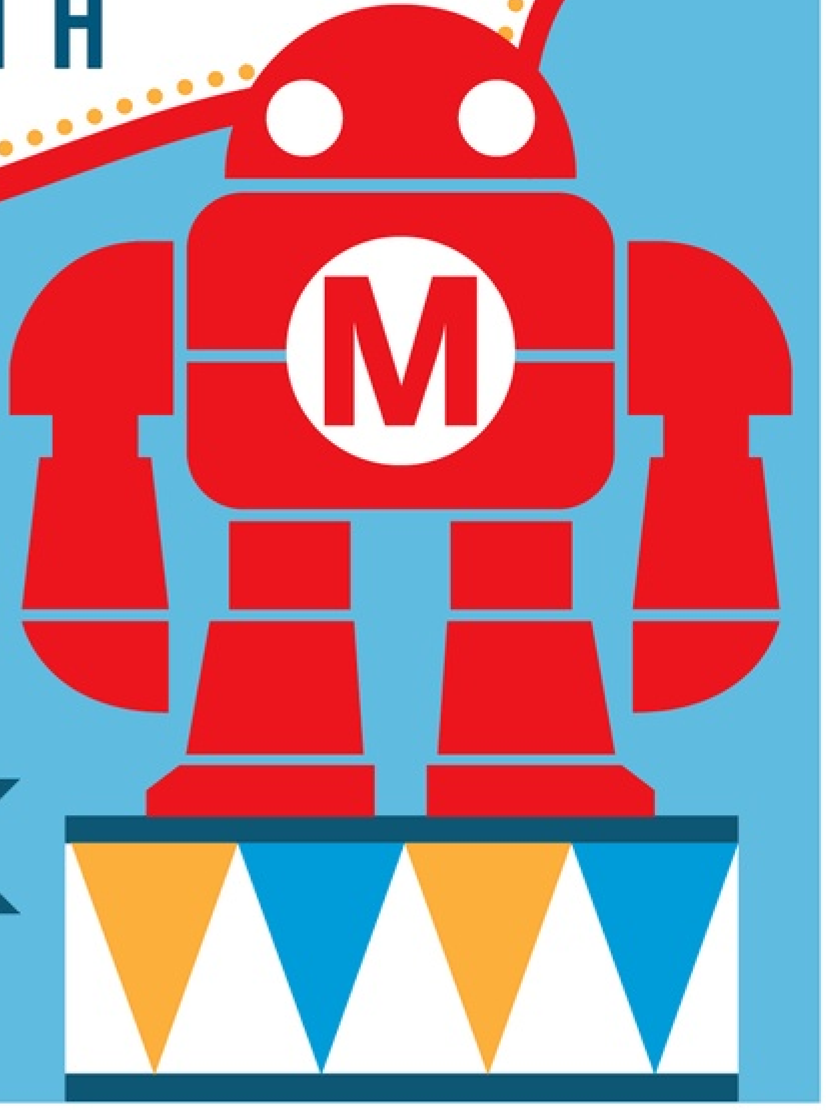
CELEBRATING *a* DECADE *of* MAKING

10th ANNUAL BAY AREA

GREATEST

SHOW & TELL

ON EARTH



MAY 16+17

SATURDAY 10AM–8PM
SUNDAY 10AM–6PM

makerfaire.com

Brought to you by
Make: magazine

Maker Faire®

PRESENTED BY intel

GOLDSMITH SPONSORS AUTODESK. DFROBOT® DRIVE THE FUTURE ITA ITALIAN TRADE AGENCY

SILVERMITH SPONSORS NVIDIA. ROBLOX

■ BY WARD SILVER N0AX
hwardsil@gmail.com

RF Oscillators

In the previous column, we learned what makes an oscillator do what it does, and tried a simple low frequency example. Now, it's time to move up — in frequency — to the oscillators which make the signals that drive the ham's wireless world.

Let's review for a moment. First, oscillators need gain (*A*), a frequency-selective filter, and a positive feedback loop (β) that all combine to satisfy the Barkhausen criterion, $A\beta = 1$.[1] Even if you didn't intend to create an oscillator, if those three conditions are present, an oscillator you will have!

At RF, the two fundamental oscillator types were devised to create feedback through reactances that formed a voltage divider with inductance (the Hartley oscillator) or capacitance (the Colpitts oscillator). That's where we left things.

Practical RF Oscillators

The two schematic snippets in the previous column showed the basic idea but weren't practical circuits, omitting such important items as power supplies, biasing, and output connections. So, how do we make an oscillator, really? **Figures 1** and **2** show a pair of actual functioning oscillator circuits.

You can build them and listen to their output signal on a world band or ham receiver between 7.5 and 8 MHz. (You'll need to listen in SSB or CW mode since the steady output has no modulation for an AM radio to detect.)

Each of the oscillators has a parallel-LC "tank" circuit that is a filter at its resonant frequency. The filter, in turn, determines the oscillator's frequency. Those components have the designators C_t and L_t.

Why is an LC circuit called a tank circuit? Because it stores energy like an electrical flywheel. The energy sloshes back and forth from the inductor to the capacitor at the circuit's resonant frequency. The positive feedback supplies a little energy and the JFET's gate takes a little out. The resonant frequency of the tank circuit is:y

$$f_o = \frac{1}{2\pi\sqrt{L_t C_t}}$$

For the Hartley circuit (**Figure 1**), $L_t = L_{t\text{-}a} + L_{t\text{-}b} = 1.48\ \mu\text{H}$ because the two inductors are in series, thus f_o = 7.96 MHz. In the Colpitts circuit (**Figure 2**), $C_t = (C_{t\text{-}a} \text{ x } C_{t\text{-}b}) / (C_{t\text{-}a} + C_{t\text{-}b})$ = 410 pF because the two capacitors are in series. Thus, f_o = 7.5 MHz. Both circuits are from an excellent electronics design reference, *Experimental Methods in RF Design* by Hayward (W7ZOI), Campbell (KK7B), and Larkin (W7PUA).

What is the purpose of the other circuit components? C_{byp} is a *bypass capacitor* to keep the drain of the

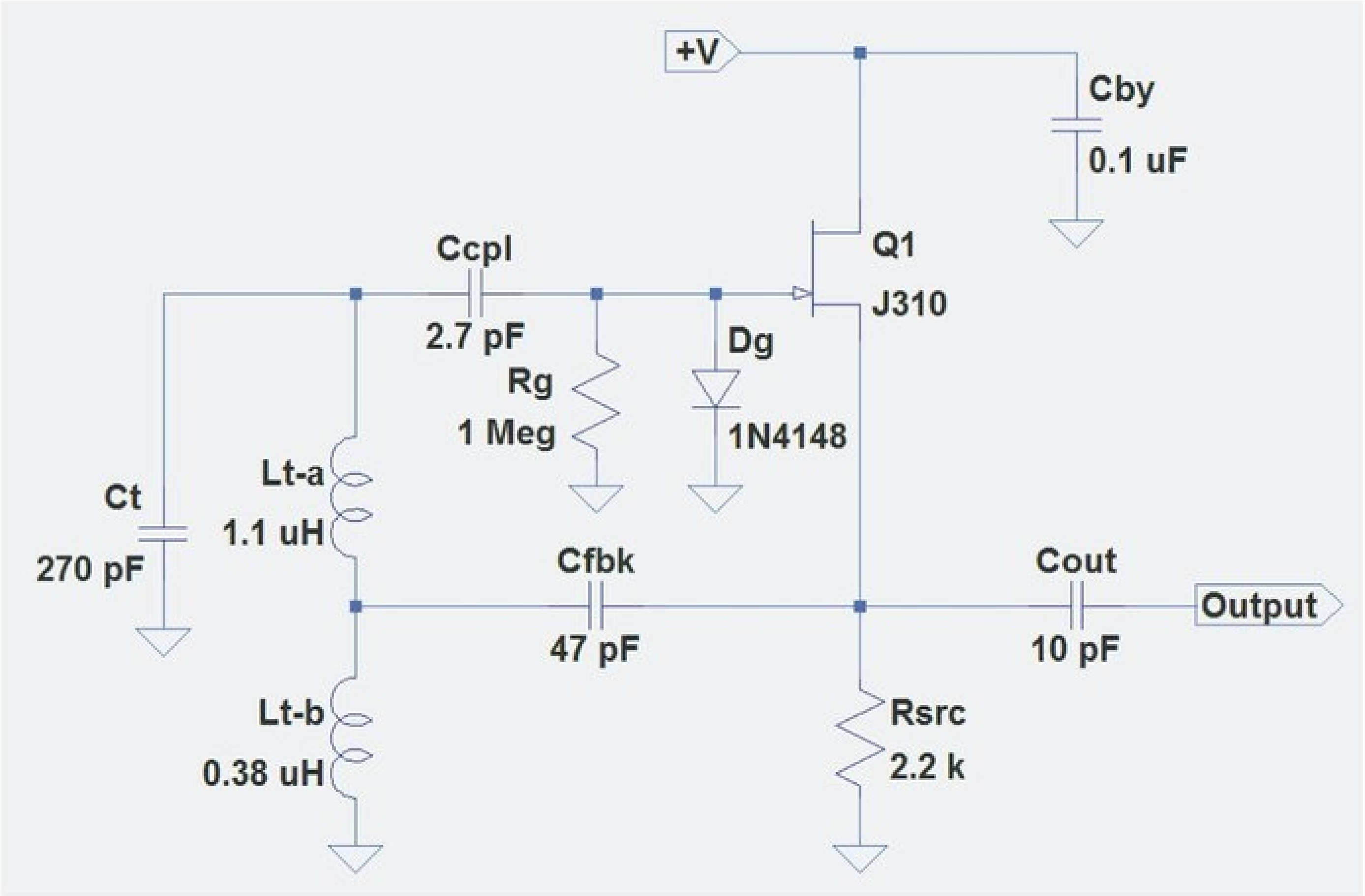


FIGURE 1. The Hartley oscillator will operate around 7.9 MHz. J310 JFETs in through-hole TO-92 packages are available from Mouser Electronics (**www.mouser.com**) and in SMT packages from many suppliers. Inductor Lt-a consists of 17 turns of #24 or #26 solid wire on a 0.5" diameter, type 6 powdered-iron toroid. Inductor Lt-b is 10 turns on a 0.3" diameter, type 6 powdered-iron toroid. For best stability, Ct should be a polystyrene film or silvered-mica capacitor. Powdered-iron types refer to Amidon Corporation products (**www.amidoncorp.com**). +V can be from 6V to 15V with excellent results.

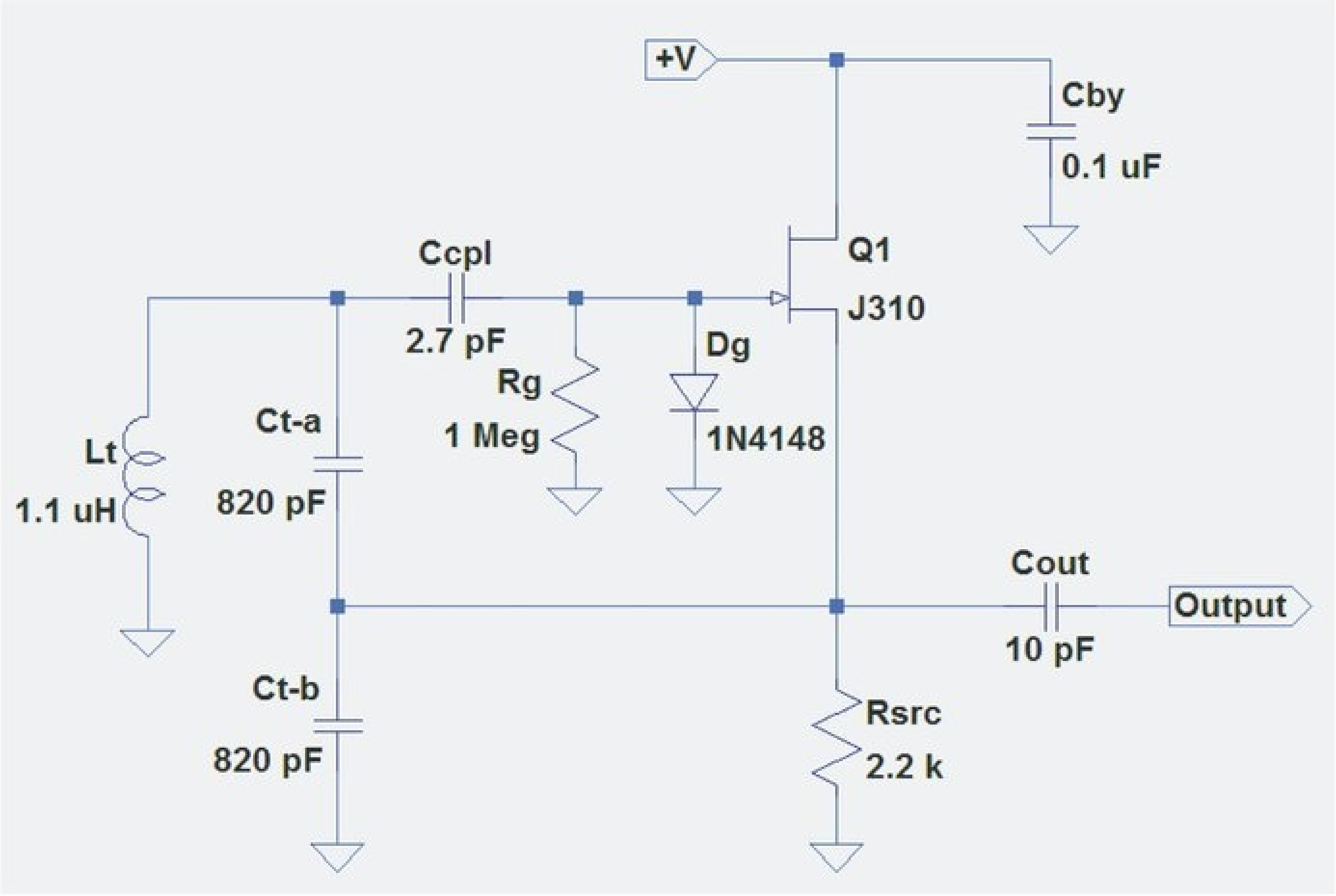


FIGURE 2. The Colpitts oscillator will operate around 7.5 MHz. Inductor Lt is the same as Lt-a in **Figure 1**, and both Ct-a and Ct-b should be polystyrene film or silvered-mica capacitors.

JFET at signal ground. This is a common drain amplifier, similar to an emitter-follower in the bipolar transistor world. R_{src} limits current through the JFET to a few mA — depending on the value of +V — which can be from six to 15 volts with good results.

Hooked on Microphonics

If you build either of the oscillators in **Figures 1** or **2**, tune them in on a receiver and let them stabilize thermally so the signal is not drifting much. Then, very very gently, tap on the circuit board or workbench near the circuit while listening to the signal. In all likelihood, the oscillator's frequency will jump with every tap.

This sensitivity to mechanical vibration is called microphonics. Any type of vibration will cause an oscillator's frequency to change. In fact, if you take the receiver into another room and have your assistant say something like, "Watson, come here, I need you!" very loudly and directly onto the oscillator, you'll probably be able to hear the oscillator's frequency being modulated by the voice.

You will also notice long-term slow changes in frequency — quicker if you bring a heat source near the oscillator's tank circuits. This is called thermal drift. It's hard to make a stable LC oscillator.

The output signal from the circuit is taken through the 2 kΩ of reactance through C_{out}. If more signal is desired, a high input impedance buffer amplifier (emitter- or source-follower) can be used to beef up the signal.

R_g stabilizes the DC voltage on the gate during oscillation, so that the JFET amplifies consistently.

C_{cpl} is a *coupling capacitor* that lets a small amount of RF leak out to the gate of the JFET, Q1. It has a high reactance at 8 MHz ($X_C = 1/2\pi fC$ = 7.4 kΩ), so the tank circuit is "lightly loaded" — meaning that the amount of energy that gets out of the tank through C_{cpl} is small compared to the energy stored in the tank circuit. This helps keep the oscillator frequency stable and reduces noise in the output signal.

C_{fbk} in the Hartley circuit is the path for feedback from the JFET's source to the tank circuit. Because it is supplying energy *to* instead of extracting energy *from* the tank circuit, its reactance can be lower (about 425 Ω). The lower value also adds less phase shift in this important signal path.

The source and gate signals for the gain-supplying JFET are in-phase, so feeding a signal back through C_{fbk} creates the positive feedback needed by the oscillator. C_{fbk} is required in the Hartley circuit to provide an RF signal path without allowing the DC current to flow through $L_{t\text{-}b}$ to ground. It is not required in the Colpitts circuit because $C_{t\text{-}a}$ and $C_{t\text{-}b}$ block any DC current flow.

D_g is a funny looking component with a purpose that is not obvious. Remember that an oscillator "starts up" by amplifying noise more and more until a self-sustaining signal is present. How does the oscillator know when to stop increasing the signal level? Well, it doesn't!

If no limiting mechanism is present, the signal will build up until it can't be amplified any further, creating a distorted square-wave-like output. Not good for radio use!

The brute force solution is D_g which begins to conduct as the positive half-cycle of the sine wave signal becomes greater than about 0.5V. That loads down the input and reduces gain, acting as a brake on the system. (Negative peaks are self-limiting since they cut off the JFET.)

You can make the oscillators in **Figures 1** and **2** adjustable — what hams refer to as a VFO which is an abbreviation for Variable Frequency Oscillator — by adding variable capacitors across the tank circuit.

A small variable capacitor of 20-30 pF across the tank circuit (from C_{cpl} to ground in either circuit) will shift the oscillator's frequency by up to 10%. Changing an inductor's value is not so easy, and clever circuit designers discovered that a variable capacitor in series with the inductor could act to cancel some of the inductive reactance, changing the oscillator's frequency as well. This is called — strangely enough — series tuning.

Effect of Component Q

Both the capacitors and inductors that determine f_o dissipate some of the RF energy flowing through them as heat. Loss in the capacitor is primarily caused by the dielectric material (such as polystyrene or mica), while the inductor loses energy to resistance in the wire and in its magnetic core. Remember that the skin effect limits inductor current to a very thin layer at the surface of the wire, so resistance at RF will be a lot higher than the resistance you measure with a DC multimeter.

The effect of these losses reduces the component's — and thus the tank circuit's — *Q*, or Quality Factor. Q can seem mysterious, but is a measure of energy loss with *Q = Energy stored during one cycle / Energy lost during one cycle*. For a component, Q is the ratio of reactance to resistance.

For example, if an inductor has 500 Ω of reactance and 5 Ω of loss resistance, it's $Q = X_L / R = 500 / 5 = 100$ — a typical value for inductors. Capacitors have much higher values of Q; several hundred and up. Higher values of Q mean the "flywheel" keeps turning without slowing down much or changing frequency. (The Q of an LC tank circuit is limited by the Q of the lossiest component — usually the inductor.)

Remember that the tank circuit acts as the primary filter for our feedback loop. The lossier the filter, the more noise it allows to get to the JFET, which happily amplifies anything that appears at the gate. Thus, in an oscillator, tank circuit Q determines the oscillator's spectral purity — meaning how much the primary desired sine wave is accompanied by noise and distortion of various sorts. If you want a clean signal, use the highest quality Ls and Cs you can.

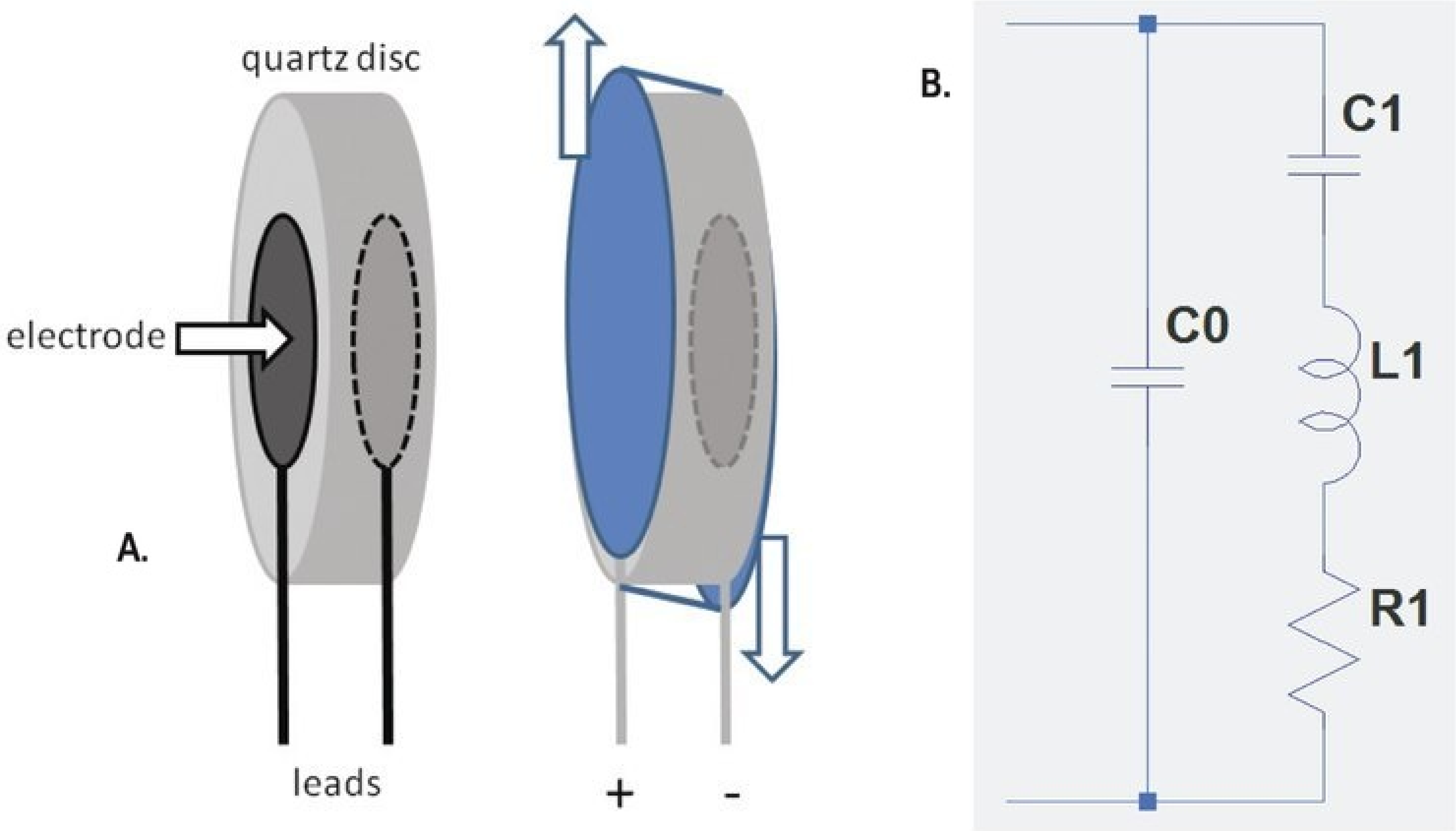


FIGURE 3. The basic construction of a quartz crystal for oscillators is shown in **A**, along with a depiction of the crystal's thickness shear vibration. The equivalent electrical circuit for the crystal is shown in **B** and described in the text.

The Quartz Crystal

Even the very best LC oscillators are not all that stable, and most have plenty of noise in their output signals. Certainly, they are handy circuits, but they are not suitable for precision jobs like generating clock signals for digital circuits and master oscillators for ham radio transceivers. In those applications, a different type of tank circuit is used: the quartz crystal.

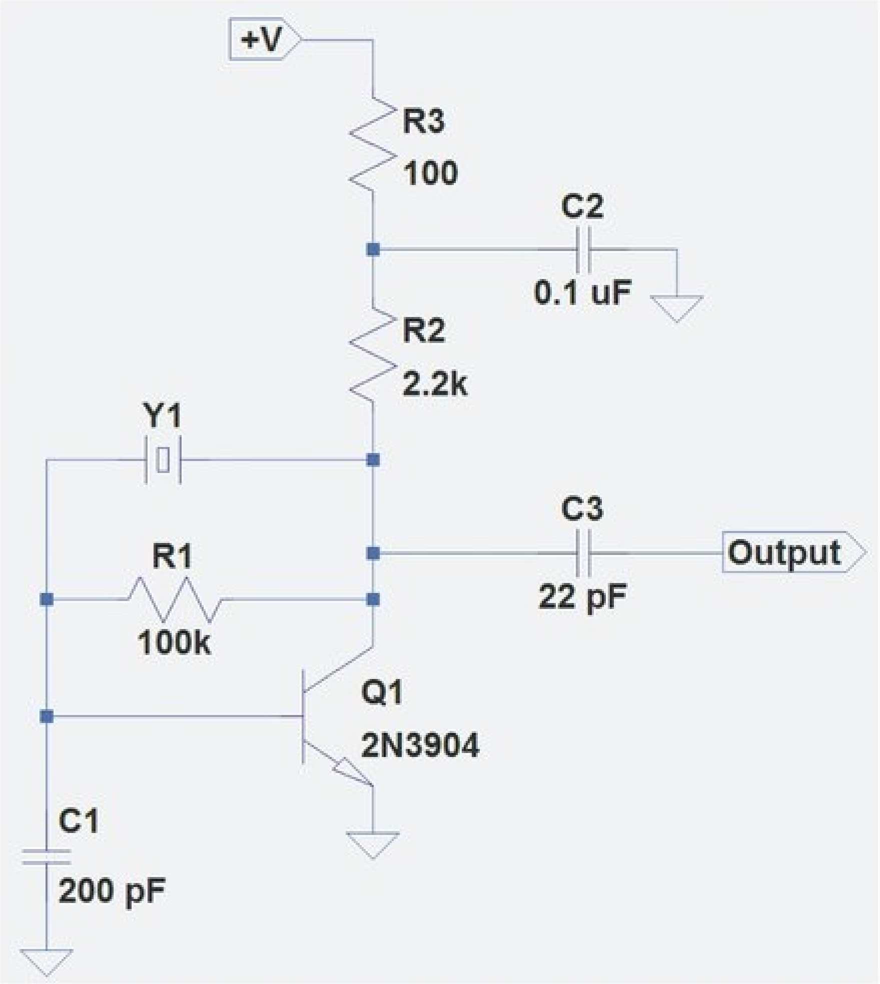


FIGURE 4. A Pierce crystal oscillator circuit that can be used from 2 to 20 MHz. If tuning of the crystal frequency is required, insert a 50 to 100 pF trimmer cap between the crystal and the transistor collector.

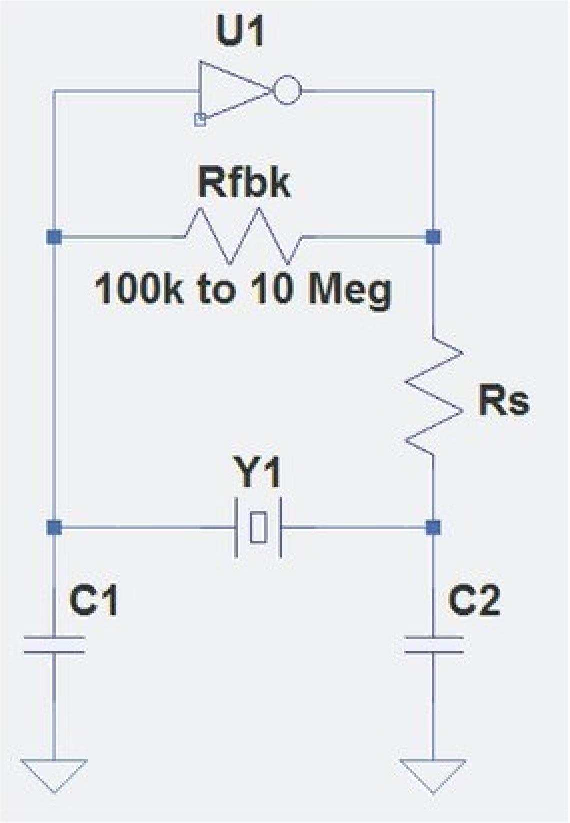


FIGURE 5. Logic inverters can be used with a crystal as shown here, providing a digital square wave output waveform directly. This type of circuit is popular with microprocessors which often provide an inverter specifically for creating a clock oscillator.

If this discussion of oscillators whets your whistle, you can find a lot of hands-on circuit building and operation among ham radio's busy and active QRP (low power) enthusiasts. Low power "rigs" (radios) are inexpensive to build and modify, and there is a lot of sharing and encouragement between "QRPers" as they design and build their gear. You can find out more about QRP operating on the ARRL's Tech Portal web page at **www.arrl.org/tech-portal** under "Technial Specialties." Click on QRP - Low-Power Operating to find a long list of clubs and web resources.

Quartz is a *piezoelectric* material in which mechanical stress creates a voltage across the material and vice versa.

Without getting into a lengthy discussion of how piezoelectricity works, imagine a thin disc of quartz with metal electrodes applied to each side as in **Figure 3A**. When a voltage is applied between the electrodes, the quartz disc deforms perpendicular to its thickness — called thickness shear — as shown in the figure. (For this type of deformation to occur, a quartz crystal must be sliced along one particular axis so that the internal molecules line up properly. This orientation of the disc is called an AT cut.)

By applying an AC voltage, standing waves can be created in the quartz disc. In return, as the standing waves move through the quartz, voltage waves are produced with the energy trading forms between mechanical and electrical. This should sound similar to how the LC tank circuit continually transfers electrical energy between the inductor and capacitor.

If the frequency of the pulses is correct, causing the disc to vibrate at its natural resonant frequency creating voltage pulses in turn, a properly configured oscillator circuit can amplify and reinforce the vibrations and voltage pulses. Just as an LC oscillator gradually builds up a steady oscillation from filtered noise, a crystal oscillator builds up the oscillation by exciting the vibrations of the quartz disc which acts as the filter.

The equivalent electrical circuit for a quartz crystal at its *fundamental* frequency — the lowest at which it has a natural resonance — is shown in **Figure 3B**. (Vibrations at higher multiples of the fundamental frequency are called *overtones*.) C1 and L1 represent the motion of the crystal as it vibrates. R1 represents the equivalent series resistance, or ESR of the crystal (usually a few dozen ohms), and C0 represents non-motional capacitance between the electrodes, stray capacitance in the crystal holder, and so forth.

What is special about the quartz disc is that it has extremely high Q — on the order of 100,000! (By comparison, a good LC tank circuit has a Q of only 100-200.) This means an oscillator that uses a quartz crystal to control its frequency will have excellent frequency stability and very low noise. This makes crystal oscillators the choice for demanding applications like generating reference frequency signals.

The Pierce Oscillator

The most common type of crystal oscillator is the Pierce circuit shown in **Figure 4**. It is a variation of the Colpitts oscillator and uses the crystal in its *series resonant mode* to create the positive feedback from the transistor's collector to the base. When the crystal's internal L and C have equal reactances, they cancel, leaving only the series resistance. There is plenty of feedback for the circuit to oscillate.

The sharp-eyed reader may have noticed that the feedback path is from the transistor's collector which is 180° out of phase with the base. This is opposite of the circuits in

Figures 1 and **2** where the feedback was from the JFET's source which was in-phase with the gate.

The extra phase shift is supplied by the crystal which — at series resonance — has between 45° and 60° of phase shift through it, requiring the extra phase shift created by R1 and C1 to make the total phase shift of 360°. R1 also provides bias for the transistor's base.

This circuit (also from *Experimental Methods for RF Design*) is not critical and will work with most crystals from 2 to 20 MHz. Any high speed NPN transistor will probably work fine. This gadget makes a terrific flea market crystal tester if a buffer amplifier is added at the output, with a simple signal detector using an LED for a visual indicator.

There are better designs for signal stability, lower noise, etc., but this simple oscillator will get you started. If you wish to "trim" the crystal's frequency to an exact value, an adjustable capacitor of 50 to 100 pF in series with the crystal will do the job.

Since hams aren't bound to fixed frequency channels most of the time, an acceptable workaround was needed to provide good frequency stability with some flexibility. Thus, the VXO or Variable Xtal (crystal) Oscillator was developed and has been used in many low power and homemade radios to simplify circuits, while still providing a good quality signal. This topic would make great extra-credit reading!

Logic Gate Oscillator

Digital designers often use a variation of the Pierce circuit with the gain provided by an inverter as in **Figure 5**. R_{fbk} biases the inverter into its linear region so that it can act as a proper amplifier. C1 and C2 create the necessary phase shift along with R_s so that there is a full 360° phase shift around the full circuit.

Because there are many types of logic families and these circuits are used over a wide range of frequencies, I recommend that you read the excellent tutorial on logic gate oscillators from the Crystek company — a well-known manufacturer of crystals.[2] Another classic reference on logic gate oscillators is the 1974 Fairchild App Note 118 on CMOS Oscillators.[3]

More Oscillators, Baby!

This pair of columns has barely scratched the surface of oscillator types and designs. There are many variations as you will see when you start looking for them.

Nevertheless, you now know a little bit about the fundamentals of how oscillators work. I'll bet you can already feel your knowledge building up to a steady oscillation! **NV**

[1] en.wikipedia.org/wiki/Barkhausen_stability_criterion

[2] www.crystek.com/documents/appnotes/pierce-gateintroduction.pdf

[3] www.fairchildsemi.com/evaluate/application-notes/familyDetails?id=10789

NEW PRODUCTS

- HARDWARE
- SOFTWARE
- GADGETS
- TOOLS

CELLULAR REMOTE TEMP MONITORING KIT

Anaren, Inc.'s, Wireless Group has announced the official launch of its Cellular Machines product line, which sends real time sensor data over a cellular network where it can be received on mobile devices, or reviewed on desktops via a cloud server. Comprised of easy-to-install hardware and a monthly cellular access/monitoring service (as low as $14.99/month), the first mass-market Cellular Machines offering from Anaren is a standard Temperature Monitoring Kit for monitoring refrigerated assets in food service, health care, and walk-in/reach-in applications such as warehouses and supermarkets. Operated on the Verizon Wireless Network and introduced in December 2014 in synch with Verizon's ongoing regional roll-out strategy, the Cellular Machines product family is one of several initiatives Anaren has embarked upon to leverage its formidable wireless experience in the exploding Internet of Things (IoT) space. Future iterations of the system will monitor other environmental factors (e.g., light, moisture, vibration, pH, power quality, among others).

According to Mark Bowyer, Anaren's Director of Wireless Business Development, the Cellular Machines system has a battery backup and runs on the most reliable communications infrastructure there is. "There's a reason first-responders, municipalities, and consumers use their cell phones during power outages and disasters."

The Cellular Machines product line began as a private company based in Austin, TX. Founded in 2010 by Gerry Cullen, who continues to provide technical and R&D expertise in support of the product line, Cellular Machines – the company – was a user of Anaren's recently introduced Anaren Integrated Radio (AIR) modules.

For its part, Verizon Wireless likewise sees potential in the IoT (a.k.a., machine-to-machine) space and particularly Anaren's Cellular Machines unique value proposition of providing end-customers with a GOOBE (Great Out Of the Box Experience) of a complete remote monitoring solution.

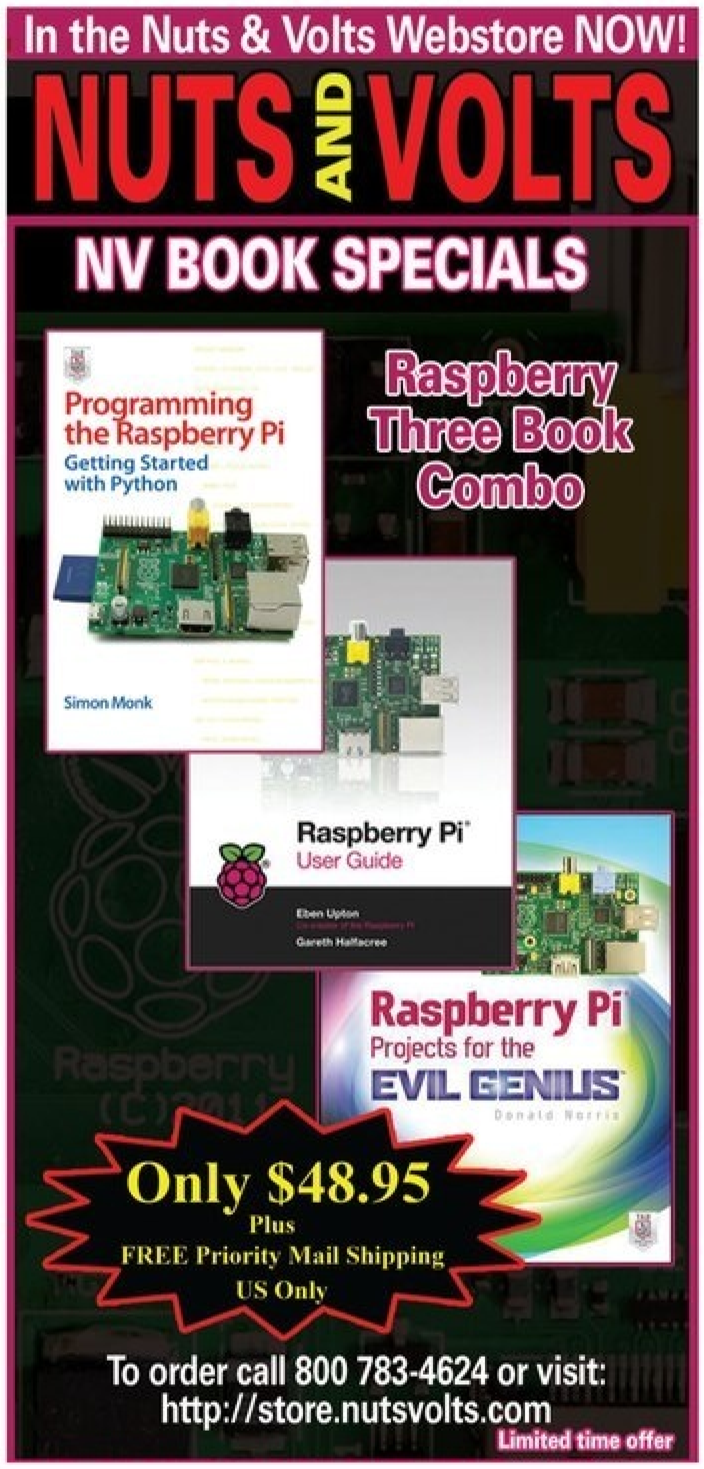
In addition to marketing its Cellular Machines product line through the Verizon Network, Anaren will also be marketing the system directly to customers via its first-ever online storefront; on a case-by-case basis, the company will also be negotiating volume and/or enterprise orders (via national accounts management arrangements), and select OEM opportunities.

For more information, contact:
Anaren, Inc.
Web: **www.anaren.com/cellularmachines**

75 GHz CLAM SHELL BGA SOCKET WITH HEAT SINK LID

Ironwood Electronics has introduced a new BGA socket design using high performance elastomer capable of 75 GHz, very low inductance, and wide temperature applications.

The GT-BGA-2019 socket is



designed for a 25x25 mm package size, and operates at bandwidths up to 75 GHz with less than 1 dB of insertion loss. The socket is designed to dissipate seven watts using a compression screw with heatsink fins, and can be customized up to 100 watts with a modified fin design on top of the screw, and by adding an axial flow fan. The contact resistance is typically 30 milliohms per pin.

The socket is mounted on the target printed circuit board (PCB) with no soldering, and uses very small real estate allowing capacitors/resistors to be placed close by. Other passive components can be placed on the back side of PCB by creating custom cutouts in the stiffener plate. The socket is constructed with a clamshell lid which incorporates a quick insertion method so that ICs can be changed out quickly. To use, place the device inside the socket, close the lid by latching, and apply downward pressure by turning the heatsink compression screw.

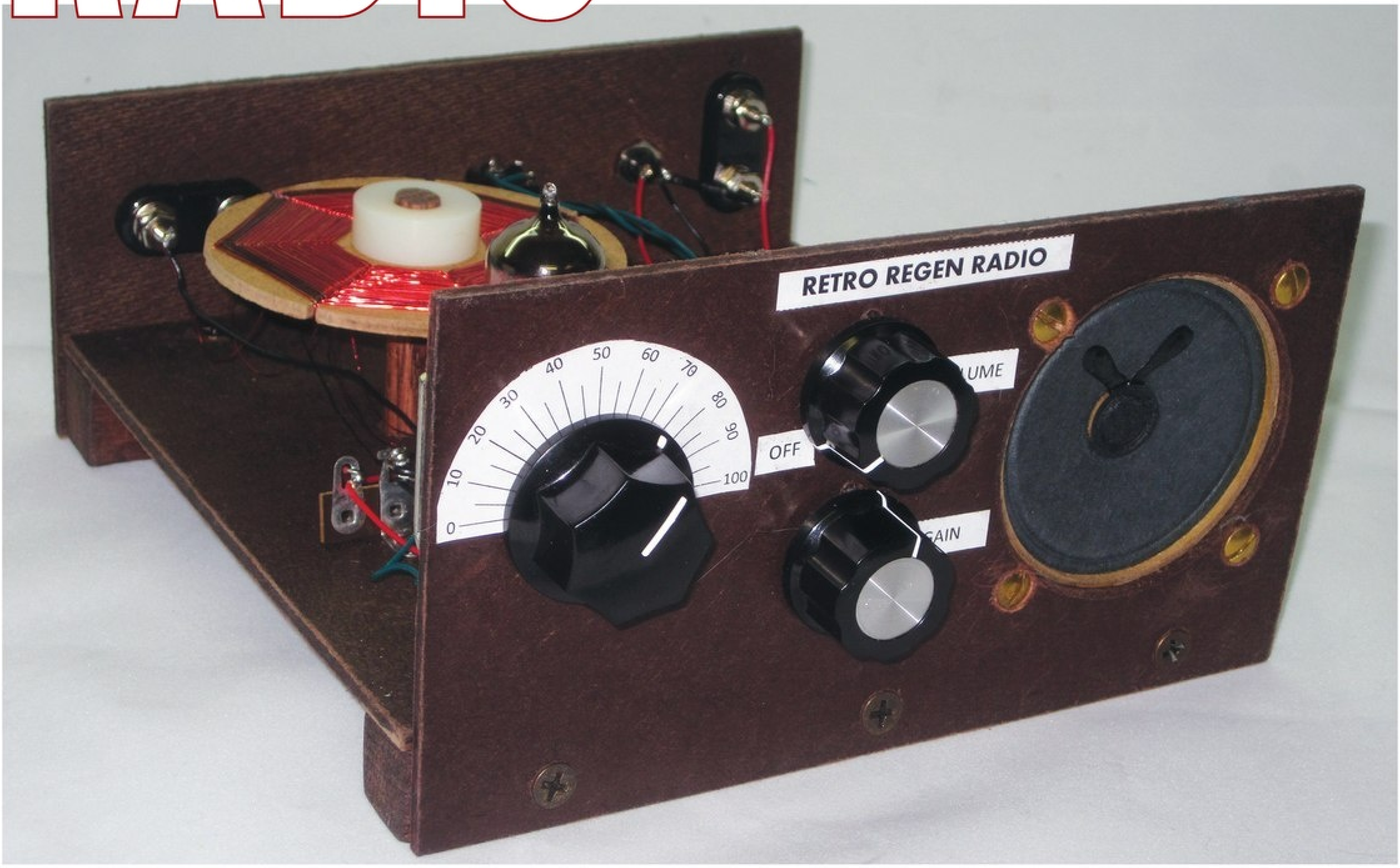
The GT-BGA-2019 socket is constructed with high performance and low inductance elastomer contactor. The temperature range is -55°C to +160°C. It works with ICs such as BGA, 25x25 mm with 24x24 array, and 1 mm pitch.

Pricing for the GT-BGA-2019 is $909 at qty 1, with reduced pricing available depending on quantity required.

For more information, contact:
Ironwood Electronics
Web: **www.ironwoodelectronics.com**

BUILD THE RETRO REGEN RADIO

By Dick Whipple



In the fast moving world of digital electronics, I find it incredible that the vacuum tube — a piece of early 20th century analog technology — has managed to survive. It should have bitten the dust long ago but that just did not happen, thanks in part to electric guitar enthusiasts and the Soviet and Chinese militaries who kept using them. The former were enamored with the sound of tube amplifiers, and the latter wanted their electronic equipment to survive a nuclear attack. Not only did the lowly vacuum tube make it into the 21st century, it is now on a noteworthy rebound. Today, many vacuum tube types are readily available and at reasonable cost. These consist of "new old stock" left over from 50 plus years ago, and many newly manufactured in modern plants world wide. A recent *NV* article describing how to revive an old tube amplifier inspired me to dust off my memories of past tube projects (some from 40 plus years back) and build a one-tube radio. The result was an exciting and fun project that I wanted to share. The radio is made using readily available parts, it operates on 12 volts making it perfectly safe, and it offers amazing performance for a simple one-tube design.

When I was growing up in the 1950s, my dad dabbled in radio/TV repair. His shop was strewn with all kinds of electronic parts. Add to this a small book of radio projects compiled from 1940's issues of *Popular Science Magazine* and I was provided with many hours of experimentation and fun. I particularly remember a shortwave design with the intriguing title "Europe on One Tube."

I am embarrassed to admit how many hours I spent trying to build radios based on these wonderful articles. Most of the designs used a regenerative circuit invented by Edwin Armstrong in 1912. A few years ago, I stumbled on an unusual regenerative design that operated on 12 volts — not the 100 or more of conventional tube designs. The radio I built turned out to be one of the best ever.

In this article, I will describe how to build and operate the broadcast band version. Should you decide to tackle it, I can promise you many hours of fun both in the building and in the listening for distant radio stations.

Vacuum Tube Background

Vacuum tube technology dates back to the time of Thomas Edison and the light bulb. In 1883, Edison noted that he could get electrons to flow between the hot filament of an experimental light bulb and a positively charged metal plate. The so-called Edison Effect only occurred in the near vacuum of the light bulb.

In 1904, the British scientist, John A. Fleming used the Edison Effect to produce the first practical tube, or "thermionic valve." Fleming's diode valve passed electrical current in only one direction, making it useful as a radio frequency detector and a rectifier for converting alternating current to direct current.

American inventor, Lee de Forest added a third element to the vacuum tube design and produced the triode, or "Audion" as he named it. He interposed a grid of wire mesh between the filament and metal plate that provided a way to control the flow of electrons.

The significant feature of his invention was that small changes of voltage on the grid would produce much larger changes of voltage on the plate, resulting in voltage amplification. Thus, a weak audio or radio signal could be amplified, which had many practical applications in telephone and radio communication.

As time went on, other advances were made in tube technology, including the addition of an indirectly heated cathode and other grids. For our purposes, the triode vacuum tube will serve as the heart of the regenerative radio receiver.

Sources

Antique Electronic Supply - **www.tubesandmore.com**
Digikey - **www.digikey.com**
Home Depot - **www.homedepot.com**
Jameco Electronics - **www.jameco.com**
New Sensor - **www.newsensor.com**
RadioShack - **http://radioshack.com**

Video introductions to soldering:
www.youtube.com/watch?v=SaUM6Q8m9iY
www.youtube.com/watch?v=I_NU2ruzyc4

Regenerative Receiver Theory

Radio detector circuits take a variety of forms. The simplest is the diode detector mentioned earlier in relation to Fleming. When the triode came along, other detectors were invented including a design called a *plate detector*. When a radio signal was applied to the control grid of a triode, detected audio could be taken from the plate circuit. The regenerative receiver takes the plate detector one step further and adds a small amount of positive feedback, resulting in "regeneration" that substantially increases circuit gain and selectivity (ability to separate nearby radio stations).

The result is a very simple circuit consisting of only one tube and a handful of components that produce amazing results. Add a couple of stages of audio amplification and you have a radio design that provides hours of fun and listening pleasure!

Circuit Description

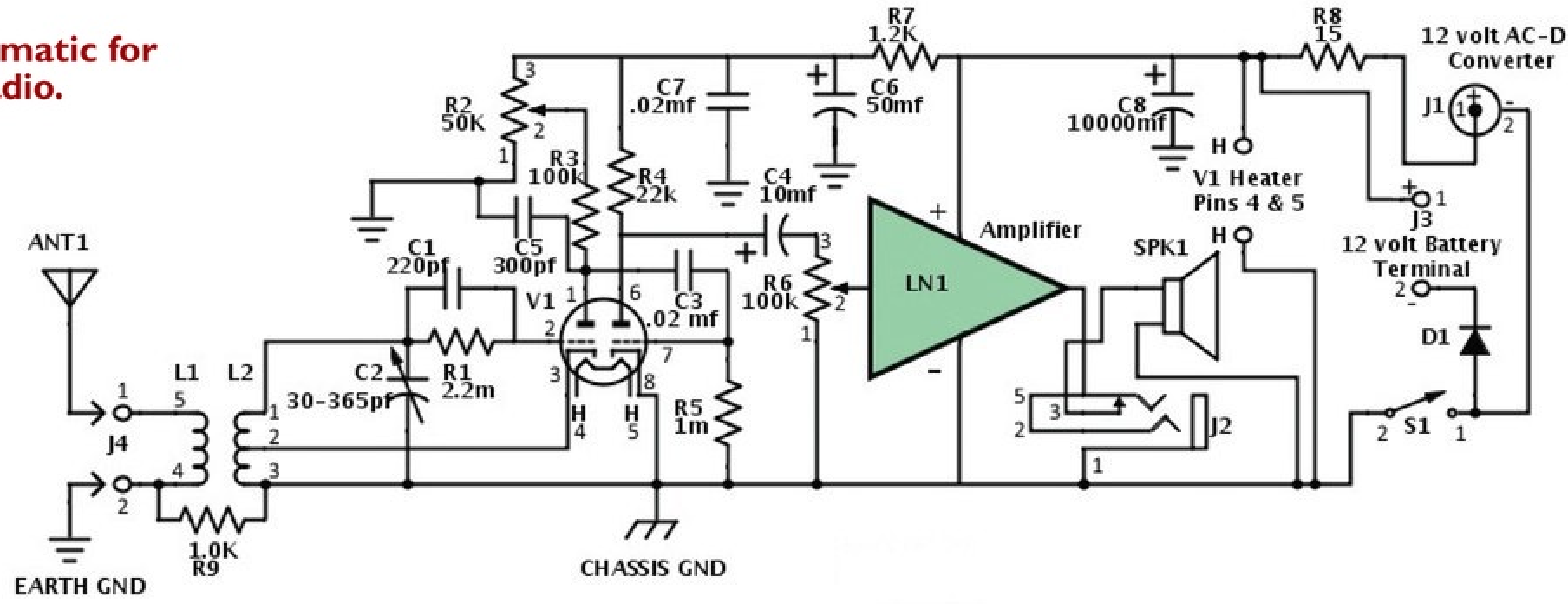
The basic circuit consists of a dual triode 12AU7. While this and other similar tubes are meant to operate at plate voltages of 90 volts or more, the 12AU7 performs amazingly well in the current application at only 12 volts. Dangerous voltages normally associated with tube projects are eliminated.

One disadvantage of low plate voltage operation is that it is not possible to develop sufficient audio power to drive a speaker or dynamic earphones. An LM386 IC power amplifier serves this purpose, making the overall design a hybrid mix of vacuum tube and semiconductor technology. The tube circuit consists of two sections: the regenerative detector and a low level audio amplifier. Refer to the schematic in **Figure 1**.

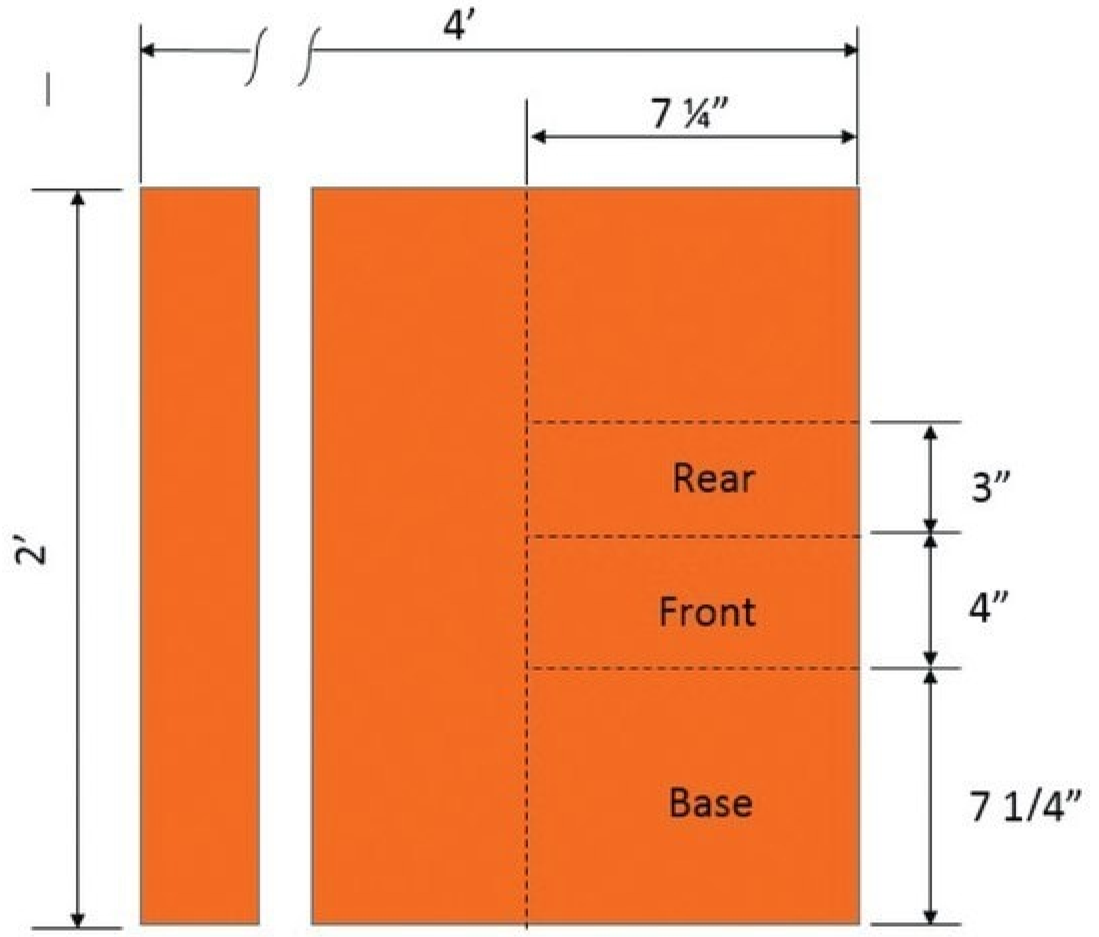
The radio frequency (RF) signal from the antenna (binding post J4) is applied to winding L1 of the spider web-wound coil. Winding L1 inductively couples the RF signal to a second winding L2 that — along with variable capacitor C1 — forms a resonant circuit covering the AM broadcast band (550 to 1600 kHz).

Capacitor C2 couples the tuned RF signal to the control grid of the triode V1-A. Resistor R1 provides a DC path to ground and "leaks" electron charge that would otherwise build up on the control grid and prevent the tube from working. A tap on winding L2 provides a small

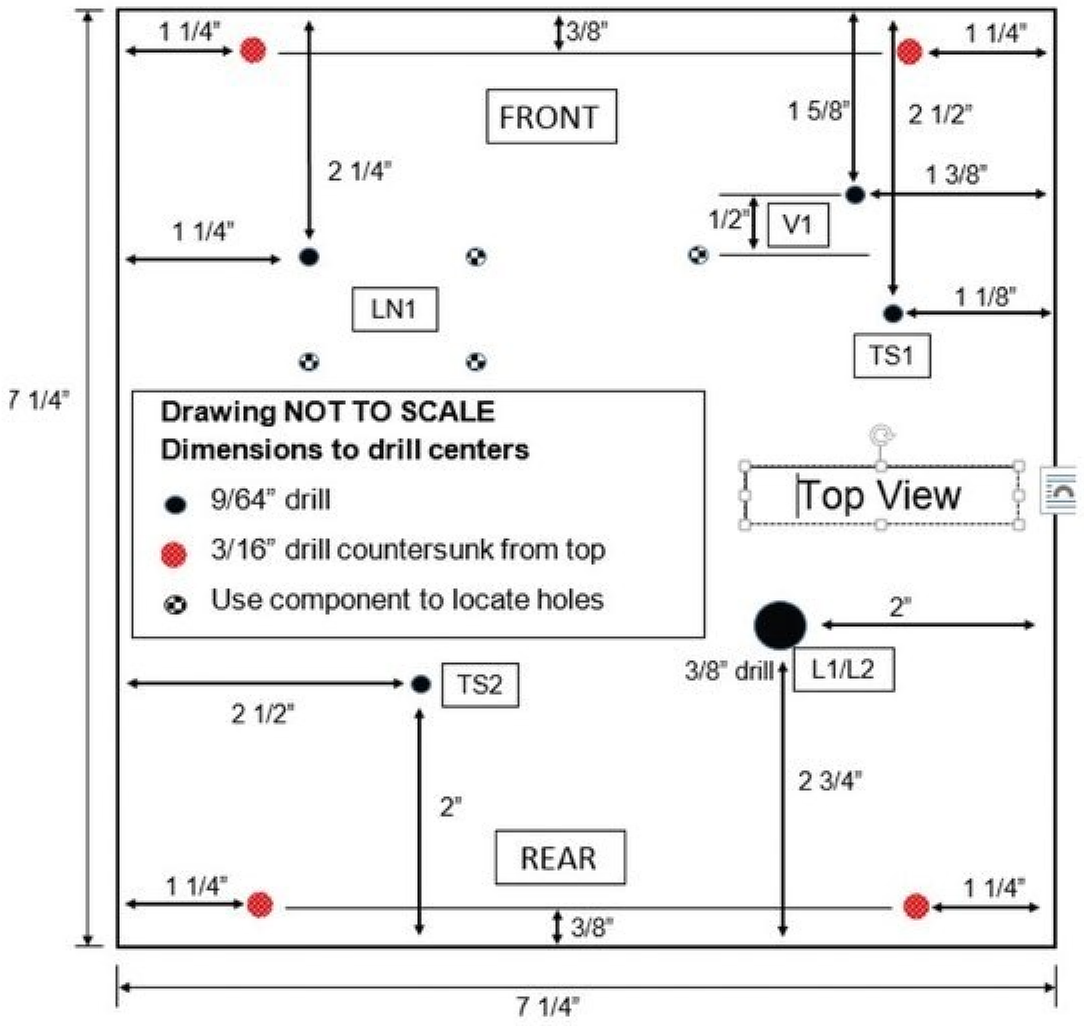
Post comments on this article and find any associated files and/or downloads at
www.nutsvolts.com/index.php?/magazine/article/may2015_Whipple.



■ **FIGURE 1. Schematic for the retro regen radio.**



■ **FIGURE 2. Hardboard sawing pattern.**



■ **FIGURE 3. Base chassis drilling template.**

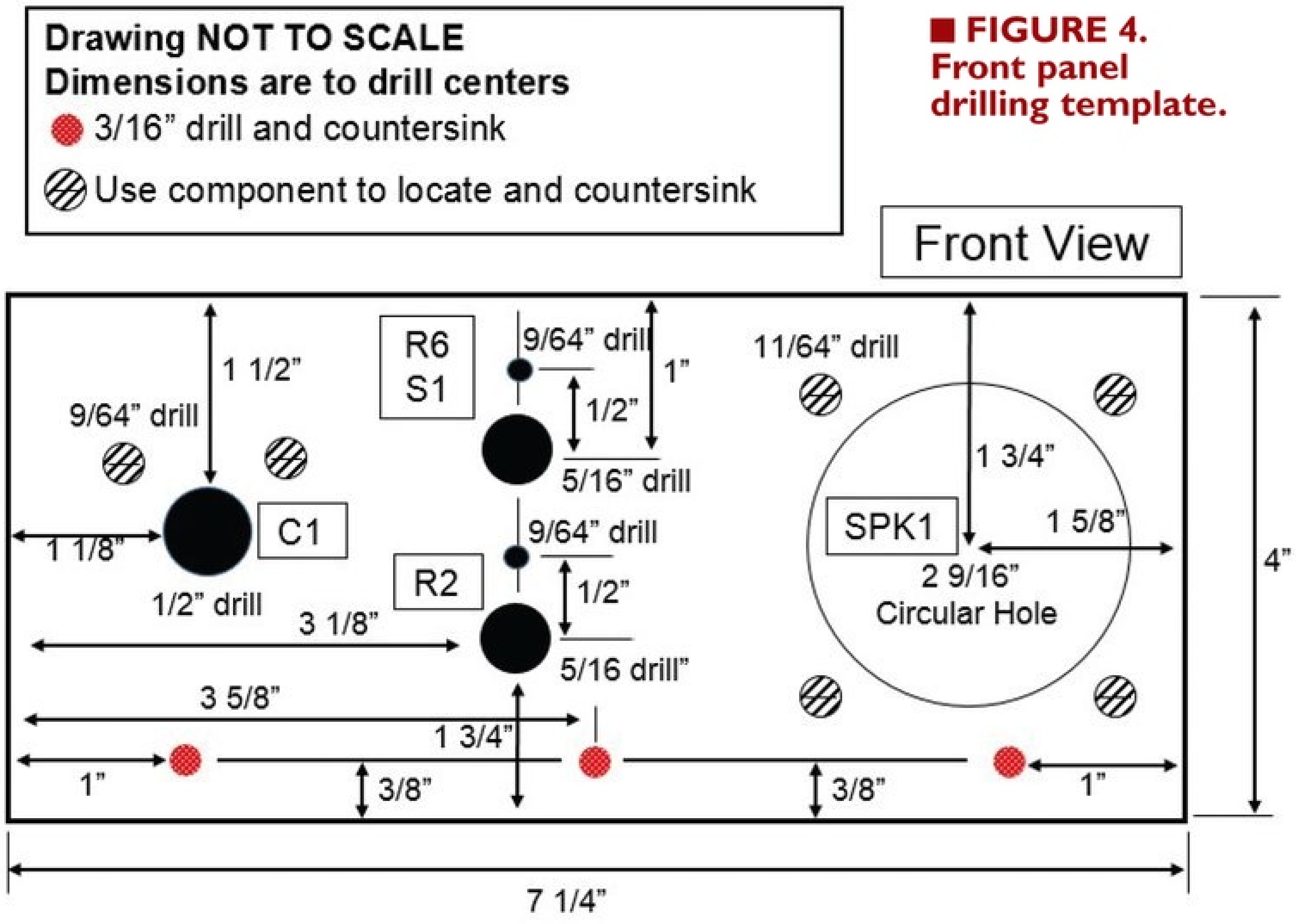
amount of positive feedback that, in turn, creates the regeneration needed to increase the gain and selectivity of the detector circuit.

Circuit gain and regeneration is controlled by varying the plate voltage of V1-A with potentiometer R3 and plate resistor R3. Capacitor C5 bypasses any remaining RF signal on V1-A's plate to ground, while C3 couples the detected audio frequency (AF) signal to the control grid of V1-B. Resistor R5 provides a grid leak path as described previously, and establishes a small reverse operating bias on the control grid. V1-B acts as a small signal audio amplifier with a gain of five. The amplified signal on the plate is coupled to the volume control R6 by capacitor C4.

From the volume control, the AF signal passes to the audio amplifier module LN-1 that boosts it to speaker volume. Earphone jack J2 is wired so that speaker SPK1 is bypassed if an earphone is plugged in. Power is provided by either a 12 volt battery (binding post J3) or an AC-to-DC power supply (jack J1). Diode D1 prevents current from flowing back into the battery should an AC-to-DC supply be plugged in at the same time as a battery. Resistor R8 and capacitor C8 provide AC hum filtering needed for the AC-to-DC power supply. Resistor R7 and capacitor C6 provide additional AC hum filtering for the more sensitive V1 circuits.

Construction and Testing

Construction is divided into three stages, namely: constructing the chassis on which the circuit will be built; wiring the electronic circuit; and finally making the spiderweb coil. Some of the construction techniques employed may be new to readers. For instance, the chassis requires basic wood working skills, and the circuit



■ **FIGURE 4. Front panel drilling template.**

is hand-wired rather than using a printed circuit board. Don't worry; I will lead you through each and every step.

Constructing the Chassis

First, we will construct the chassis. Traditionally, radio chassis are constructed from aluminum or steel. Metal working has its own set of challenges and requires specialized tools, like expensive chassis punches. I chose instead to use tempered hardwood as a base for mounting components. Interesting, some of the earliest radios were built this way. (Refer to the image at **www.duanesradios.info/html/scott_superheterodyne.html**.)

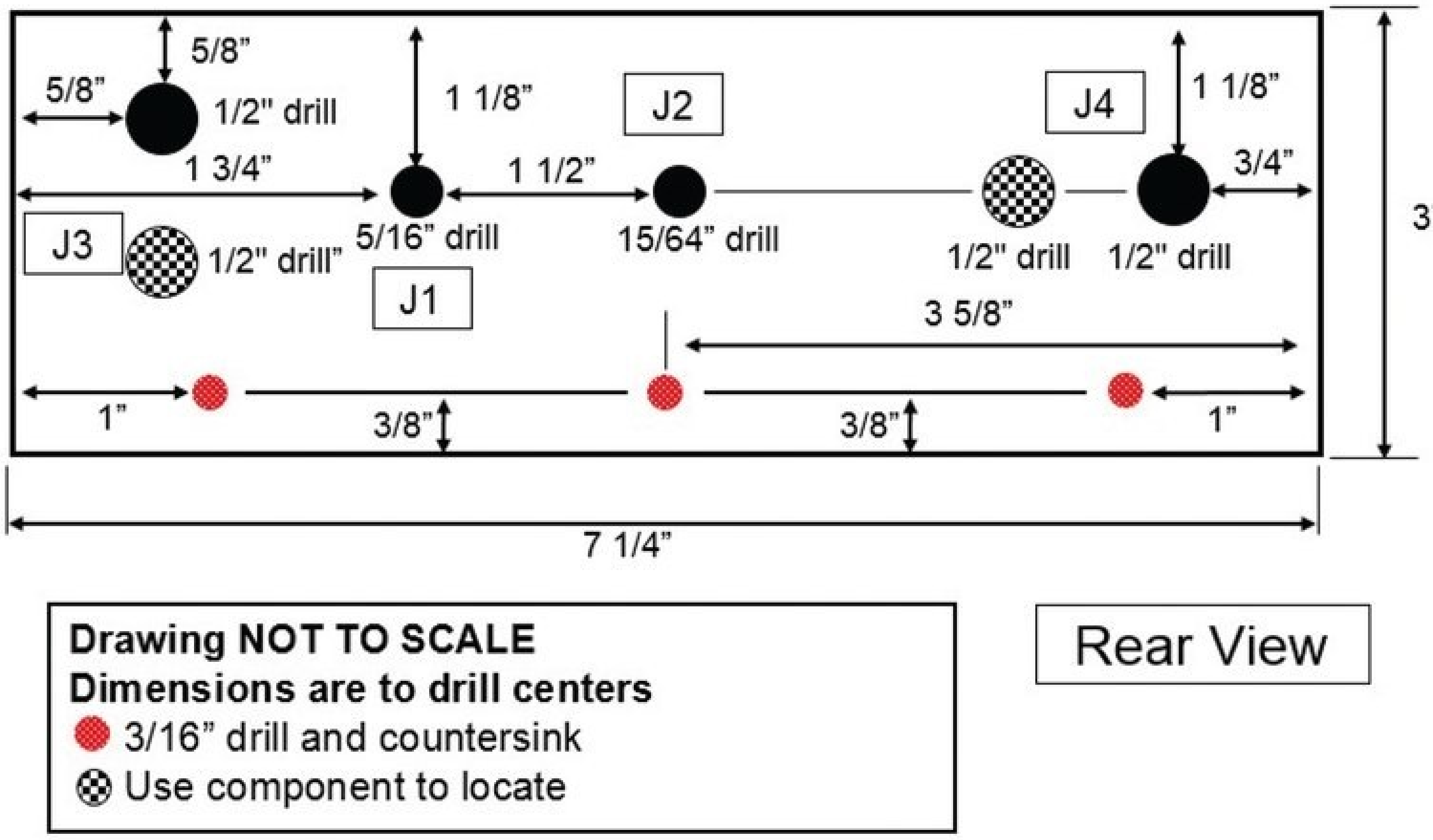
Start with a 2' x 4' piece of 1/8" tempered hardwood. Cut a 7-1/4" strip across the 2' width, then cut this piece at 7-3/4", again at 4", and finally at 3" (see **Figure 2**). These pieces are, respectively, the base, front, and rear assemblies of the chassis.

To locate holes simply to be drilled, trim quad-ruled graph paper (four squares per inch) to fit each chassis assembly. Spray the reverse side of the graph paper lightly with spray adhesive and place the graph paper so that it lines up exactly with edges of the finished side of the hardboard. Press the graph paper down from the center out to remove an air bubbles and obtain a smooth result.

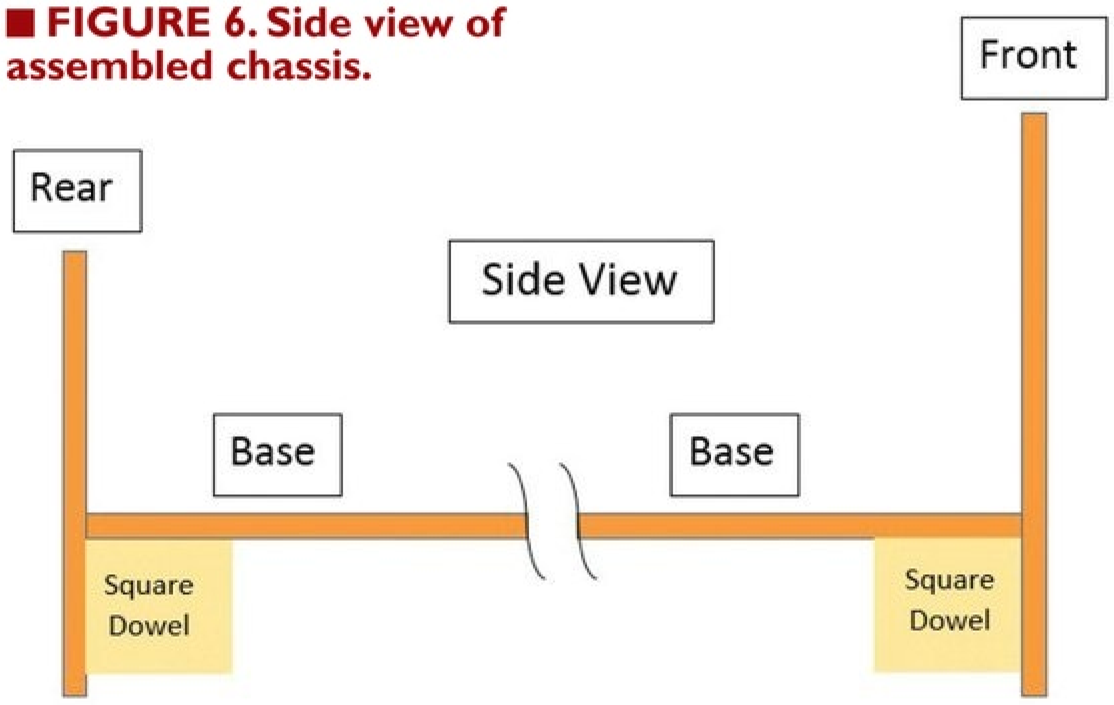
Use the layout drawings (**Figures 3**, **4**, and **5**; files are available at the aticle link. Graphics shown here are for reference only.) to mark the drill locations and drill sizes on the graph paper. Note that holes for the audio module LN1, V1, C2, and binding posts J3 and J4 are located using the actual component to ensure correct hole location. Use an awl or ice pick to precisely locate where each hole will be drilled. If you plan to use an electric hand drill, pre-drill 1/16" pilot holes, then use the drill size specified. Clean up the holes by rubbing gently with medium sandpaper.

The circular hole for the speaker will require a hole saw. Cut a 2-1/2" diameter hole first, then widen as necessary by sanding the circle edges so that the speaker rim gasket fits snugly into it. With the speaker in place, mark and drill its mounting holes.

Variable capacitor C2 also requires special handling. Place the shaft rim in the 1/2" drilled hole and note the two threaded holes on the front of the capacitor. From inside the capacitor frame, use a sharp pencil to mark the holes. Drill the holes and check that they line up correctly.



■ **FIGURE 5. Rear panel drilling template.**



■ **FIGURE 6. Side view of assembled chassis.**

Jacks J1 and J2 have thread lengths too short to reach through the 1/8″ thick hardboard.

A simple solution is to use a flat tip 1/2″ drill bit to carefully remove enough material from the back of the rear assembly to allow the threads to protrude sufficiently out the front side.

After all the holes are drilled, peel off the graph paper and clean the surfaces. This is a good time to apply a dark walnut stain if you desire.

Cut two 7-1/4″ lengths of square dowel, then use 12 brass wood screws to attach the front and rear assemblies to the base. Use 1/8″ pilot holes into the dowel to avoid splitting it (see **Figure 6**). Mount all the components to the assembled chassis except for audio module LN-1 and the V1 socket. When satisfied with the fully assembled and populated chassis, remove the front and rear assemblies from the base.

Leave the base attached to the square dowels. Chassis construction is now complete.

Wiring the Circuit

The next stage of construction involves wiring the circuit. Mount the socket for V1 *upside down* on the base, using 3/4″ standoffs and 6-32 x 1″ machine screws. This will provide a convenient platform on which to pre-wire components associated with V1. If you are not experienced with soldering, search YouTube for "how to solder." You can also refer to the recent series, "Basics of Soldering," which started in the December 2014 issue of *SERVO Magazine* (**www.servomagazine.com**).

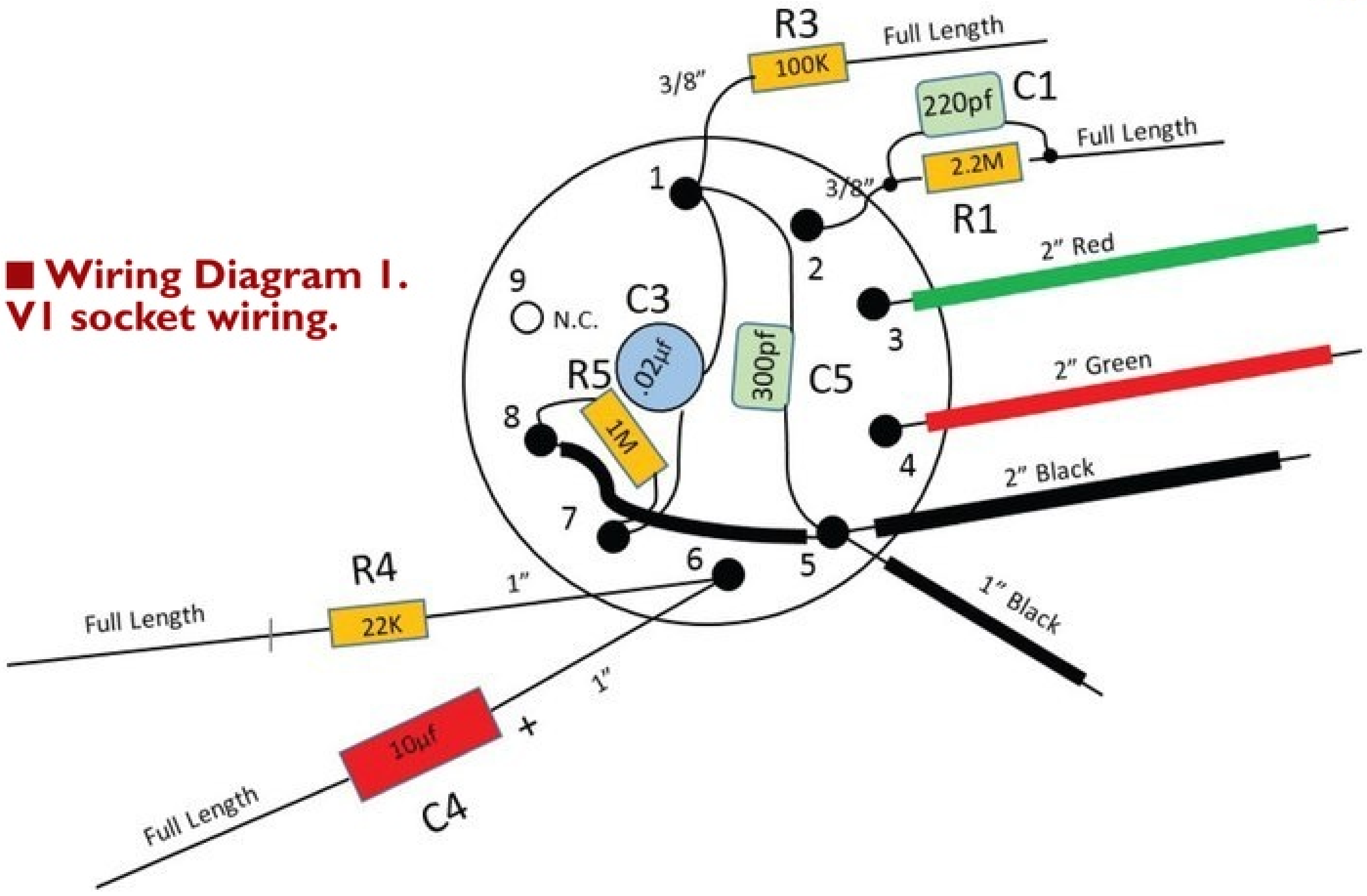
Use **Wiring Diagram 1** to wire components connected to V1's socket. Wire the components in this order: connect the wire from pin 5 to pin 8 first; then connect R5, C3, and finally C5, layering one above the other. The remaining components and wires can be added in any order.

Follow the recommended lead lengths, allowing 1/4″ additional to wrap around the connecting terminal for mechanical stability. For instance, if the lead length specified is 3/8″, cut the lead initially to 5/8″ (3/8″ + 1/4″). Use spaghetti wire insulation on all bare leads longer than 1/4″. Solder all connections.

When done, inspect all soldering joints, then remount V1's socket right side up.

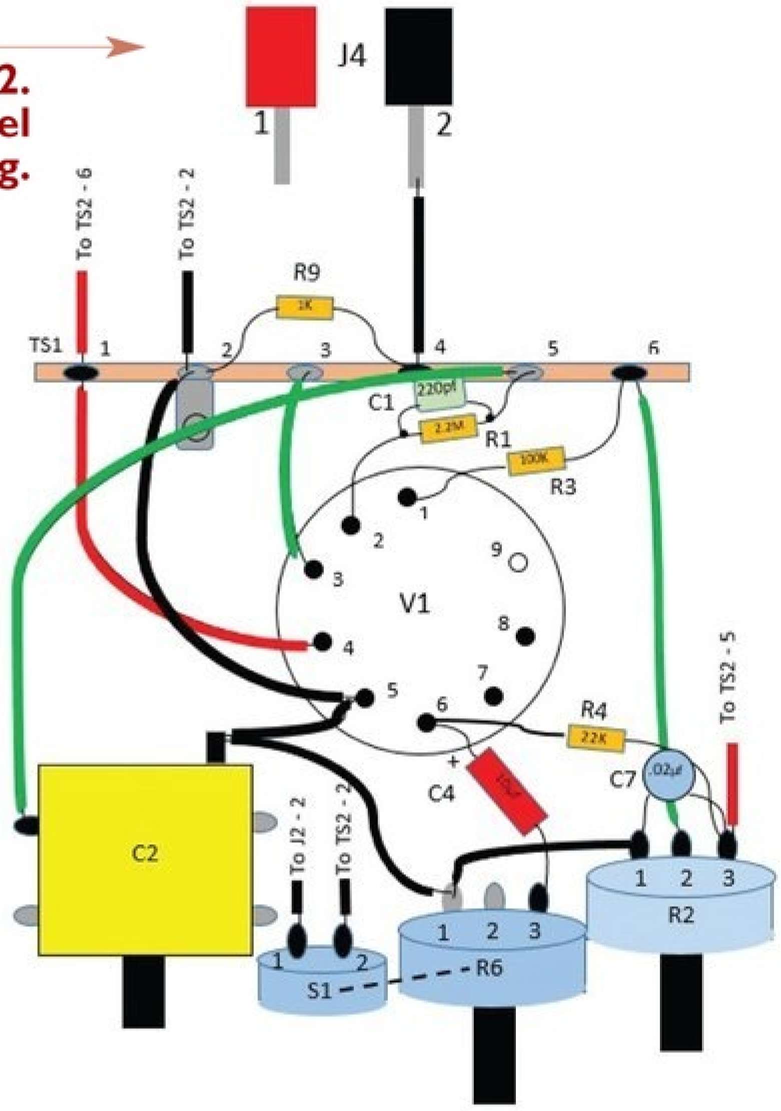
Before moving to the next stage of wiring, mount the front and rear assemblies to the base. Use **Wiring Diagram 2** to connect the wires and components previously prepared on the V1 socket.

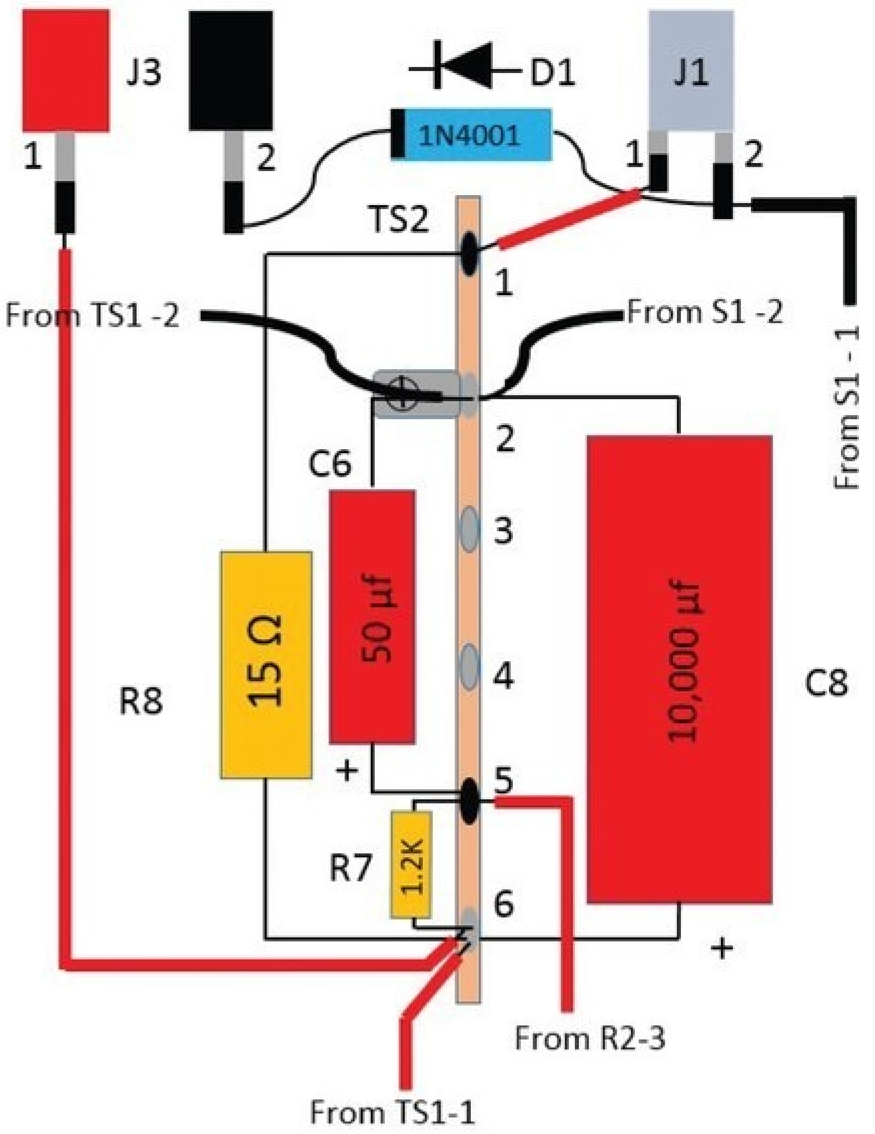
Now, add C7, R9, and the additional wires including those to terminal strip TS2 and J4. Trim wires and leads to their minimum length and use spaghetti wire insulation on



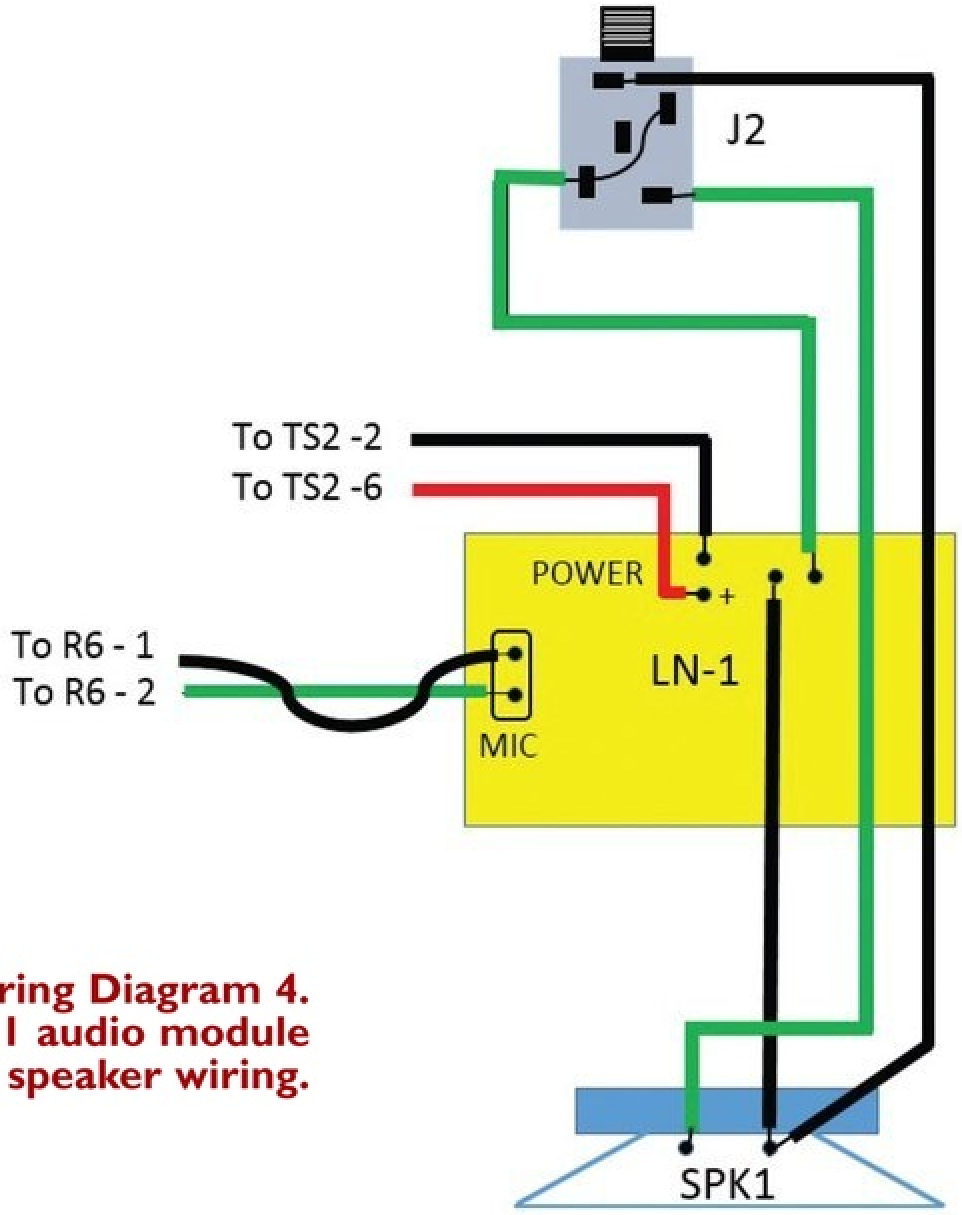
■ **Wiring Diagram 1. V1 socket wiring.**

■ **Wiring Diagram 2. V1 and front panel wiring.**





■ **Wiring Diagram 3. Power supply wiring.**



■ **Wiring Diagram 4. LN-1 audio module and speaker wiring.**

all bare leads longer than 1/4". When wiring is complete, solder the connections shown filled in with black; leave the gray connections for soldering later.

Use **Wiring Diagram 3** to wire the power supply components. Trim wires and leads to their minimum length and use spaghetti wire insulation on all bare leads longer than 1/4".

When wiring is complete, solder the connections shown filled in with black; again, leave the gray connections for soldering later.

The next step is to build LN-1: the audio amplifier module. Instructions are included with the kit. A few changes must be made. Do not install the microphone or the 3.3K ohm resistor (LN-1 R1).

Also, replace the 1K ohm resistor (LN-1 R2) with a 100K ohm, and the 10K ohm resistor (LN-1 R3) with a 680K ohm. The change is necessary to decrease LN-1's loading effect on V1-B's output.

Add external wires to LN1 as indicated on **Wiring Diagram 4**, adhering to the color codes. Make all wire lengths 6" initially. Note that the input wires connected to the "MIC" point are tightly twisted for the first 3-1/2". Adjust R6 (the LN-1 gain control) to maximum, fully counter-clockwise.

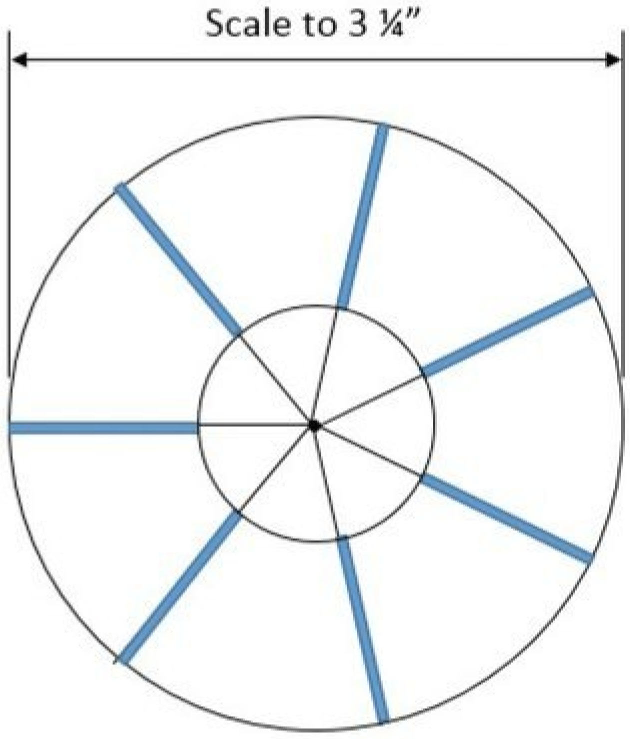
After LN-1 is built, mount it to the base with 1/4" standoffs and 4-40 x 1" machine screws. Use **Wiring Diagram 4** to wire audio module LN-1 to V1, the power supply, and the speaker/earphone circuit.

When wiring is complete, solder all connections.

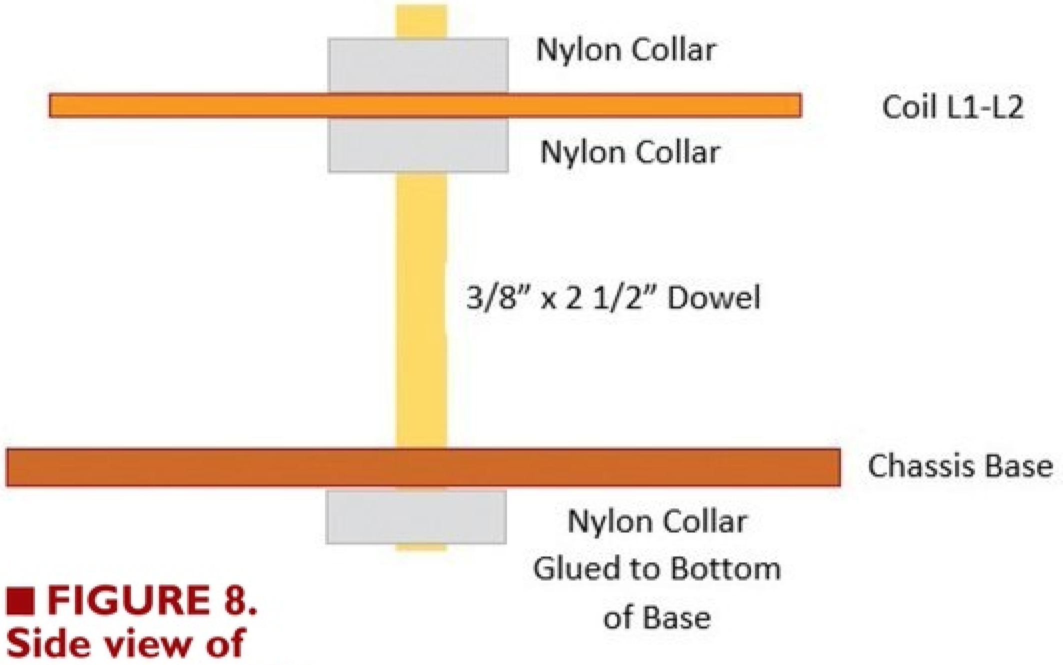
Constructing the Spiderweb Coil

The final stage of construction involves making and installing the spiderweb coil. The fiberboard used as backing for picture framing is an excellent choice from which to make the coil. You may have to search around to find just the right material. The perfect selection will be slightly less than 1/8" thick and similar to (but not as hard as) the tempered hardboard.

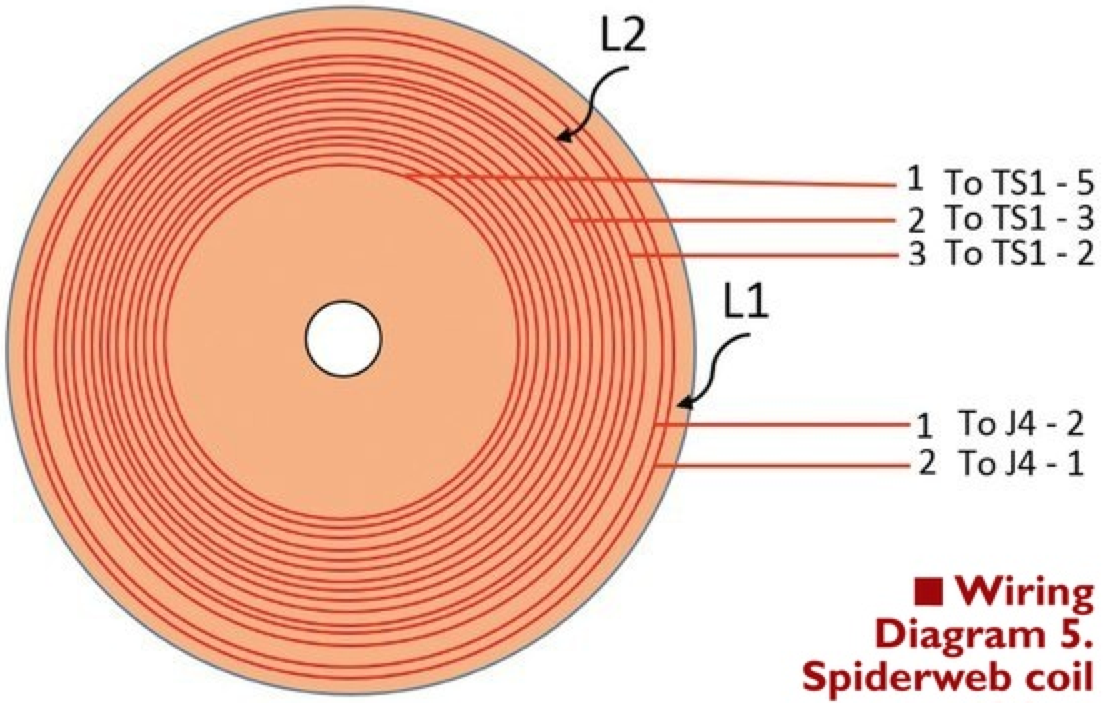
Copy the pattern of **Figure 7** and glue it to the fiberboard. Use a 3-1/2" circular saw to cut out the coil shape. The finished coil will be nearer the 3-1/4" diameter



■ **FIGURE 7. Template of spiderweb broadcast band coil.**



■ **FIGURE 8. Side view of spiderweb coil assembly.**



■ **Wiring Diagram 5. Spiderweb coil wiring.**

of the pattern. Use medium grade sandpaper to clean up the edges.

While holding the coil in a vice, use a hack saw to cut the seven slits. After each slit is sawed, use medium sandpaper to clean it out and round the edges of the cut. This is most easily done by folding the sand paper in half and passing it back and forth within the slit.

Carefully check that the depth of each slit is the same. The circular saw will have made a hole in the center of the coil. Install a 1/4" x 2" long screw in the hole with the head against the smooth side. Secure it with a nut. You will wrap the coil leads around the extended threads of the screw to keep them out of the way while winding the coil.

To make winding L2, wind 5" of #28 enameled wire around the screw, then pass the remaining length through a slit. This will be lead 1. Pull it tight on the other side and pass it down through the next slit. Repeat this until you have made about 65 complete turns. Note that counting the wire turns on either side represents roughly half the total number of turns.

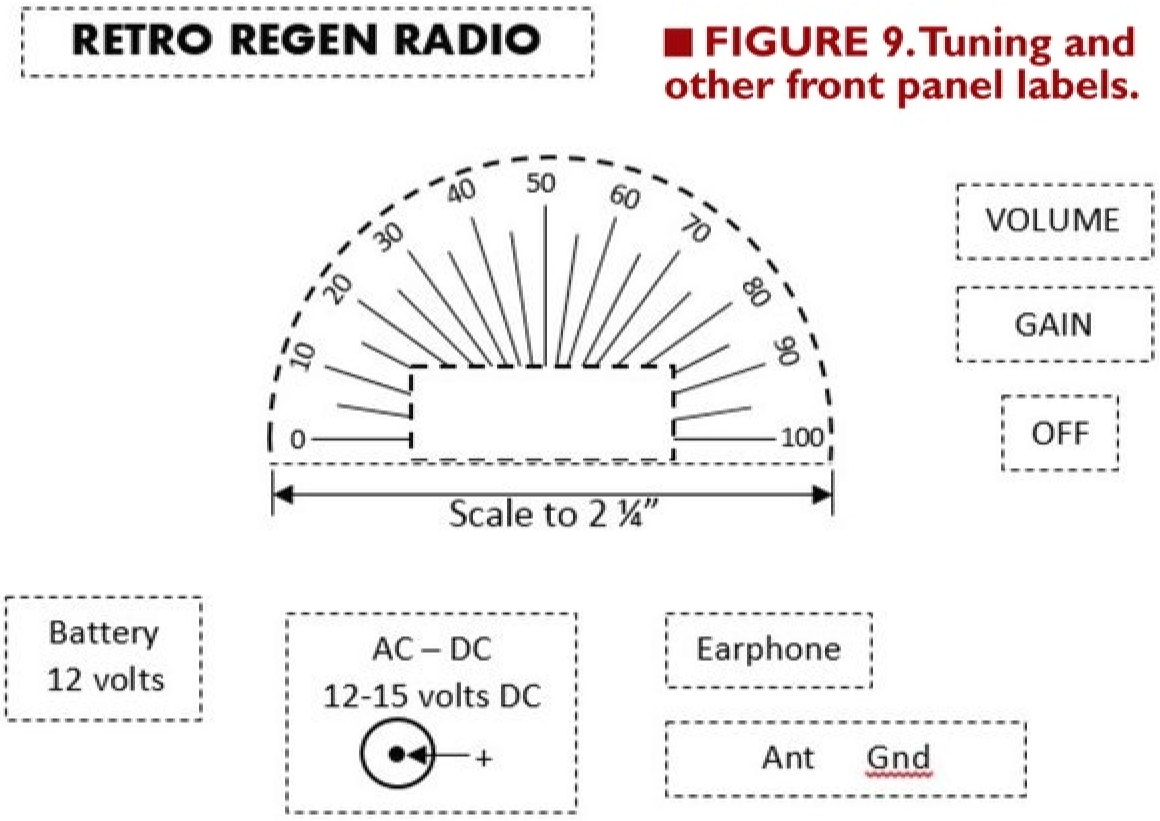
End winding with the wire passing back through the slit where you started. Fold the next 10" of wire in half, wrap it around the screw with lead 1, and pass the remaining wire through the same slit. The folded wire will be the coil tap lead 2. Continue winding in the same direction for about 13 turns, finishing at the same slit as before. Cut the wire to 5" and wrap this lead around the screw. This will be lead 3. Winding L2 is complete.

Leaving 5" leads, follow the same procedure for L1, starting and ending where you finished the L2 winding. Wind about five turns in the same direction. The starting lead is number 4 and the ending lead is number 5. When all windings are complete, dab a little quick set epoxy at the outer edge of the slit to keep lead 5 from unraveling.

Unwind the leads from the screw and gently pull them aside. Drill out the center hole to 3/8". Cut a 3-1/2" length of 3/8" round dowel. Insert one end in an electric drill and sand the rotating dowel evenly until the nylon collars fit snugly over it. Cut 2-1/2" of the sanded portion and set it aside; refer to **Figure 8**.

Glue a collar to the underside of the chassis base to hold the dowel firmly in place. Use **Wiring Diagram 5** to connect the coil leads to terminal strip TS-1 and binding post J4. Solder all connections. Insert the dowel through the chassis base into the glued collar. Slip one collar on the dowel, then the coil (lead side down), and a second collar to hold the coil in place as shown in **Figure 8**.

Use spray-on glue to attach the tuning scale and labels of **Figure 9** to the front assembly. Lastly, install the knobs to the front of the radio. This completes construction.



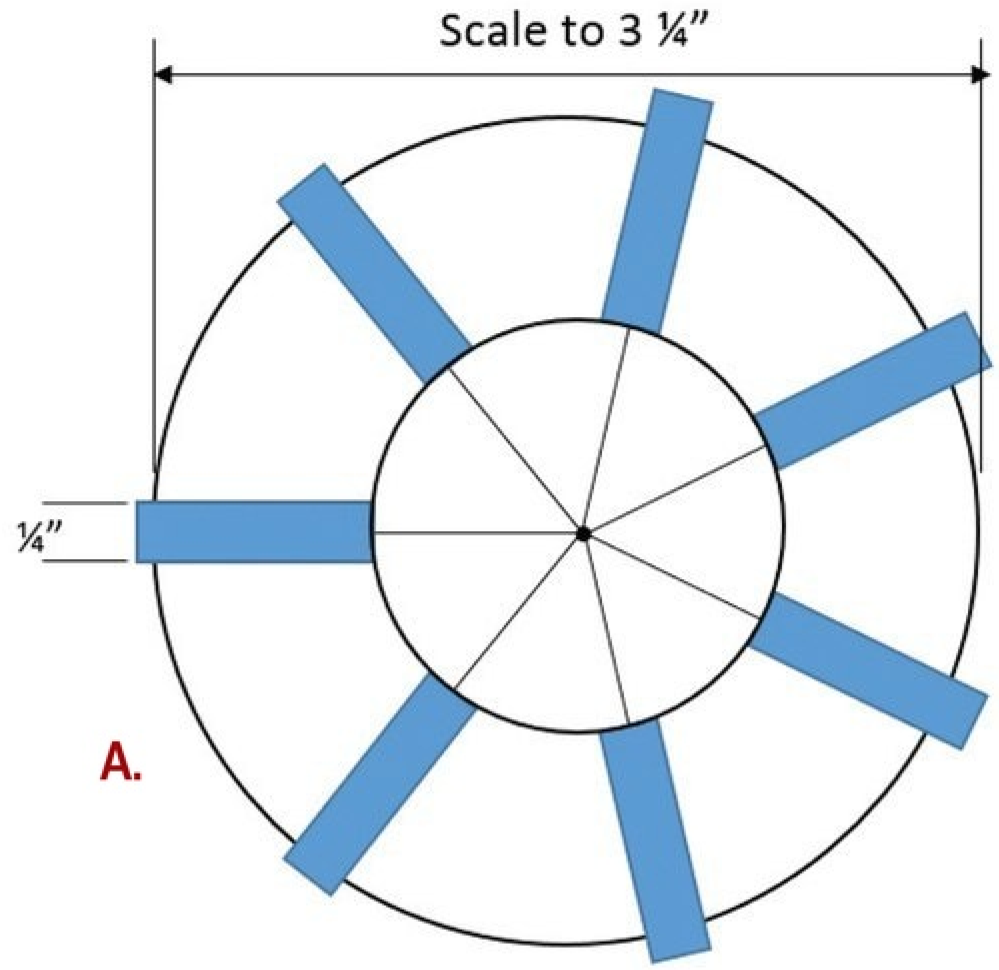
■ **FIGURE 9. Tuning and other front panel labels.**

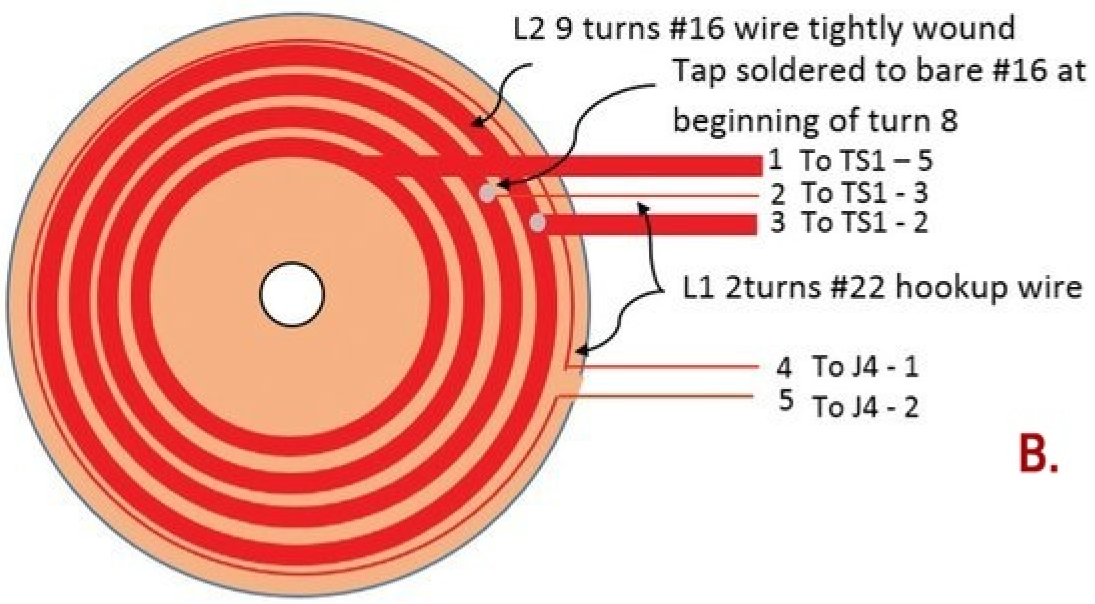
Preliminary Testing

Before applying power the first time, it is a good idea to check for a major short circuit. Before inserting

PARTS LIST

ITEM	COMPONENT	DESCRIPTION	SOURCE	PART#
C1	220 pF @ 500 VDC	Silver Mica Capacitor 5% Radial Lead	Antique Elec	4957-220
C2	30-365 pF	Variable Capacitor	Antique Elec	5317
C3	.022 mF @ 50 VDC	Disc Ceramic Capacitor 20%	Jameco	15245
C4	10 mF @ 50 VDC	Axial Capacitor 20%, 85C	Jameco	10882
C5	300 pF @ 500 VDC	Silver Mica Capacitor 5% Radial Lead	Antique Elec	4957-300
C6	50 mF @ 50 VDC	Axial Capacitor 20%, 85C		11105
C7	.022 mF @ 50 VDC	Disc Ceramic Capacitor 20%	Jameco	15245
C8	10,000 mF @ 16 VDC	Axial Capacitor 20%, 85C	Jameco	93681
D1	1N4001	Diode Rectifier	RadioShack	2761101
J1	Power Connector	Connector, Power, PC712A	Jameco	297553
J2	Phone Jack	Stereo 2.5 MM, Tip Switch	RadioShack	2740246
J3	DC (Battery) Supply	Binding Post - Dual Banana Red and Black	Jameco	125197
J4	Antenna Terminal	Binding Post - Dual Banana Red and Black	Jameco	125197
L1	Coil - Primary	*See text.*		
L2	Coil - Secondary	*See text.*		
LN1	Amplifier	Kit, SUPER SNOOPER - BIG EAR	Jameco	151204
R1	2M ohm	1/2 watt Carbon Film Resistor	Antique Elec	3647-2M
R2	50K ohm Potentiometer	Linear 0.5 watt with Switch (S1)	Jameco	255549
R3	100K ohm	1/2 watt Carbon Film Resistor	Antique Elec	3647-100K
R4	22K ohm	1/2 watt Carbon Film Resistor	Antique Elec	3647-22K
R5	1M ohm	1/2 watt Carbon Film Resistor	Antique Elec	3647-1M
R6	100K ohm Potentiometer	Linear 0.5 watt with Switch (S1)	Jameco	263822
R7	1.2K ohm	1/2 watt Carbon Film Resistor	Antique Elec	3647-2M
R8	15 ohm	1 watt Carbon Film Resistor	Antique Elec	3649-15
R9	1K ohm	1/2 watt Carbon Film Resistor	Antique Elec	3647-1K
	100K ohm	1/2 watt Carbon Film Resistor - *Replace R2 LN1*	Antique Elec	3647-100K
	680K ohm	1/2 watt Carbon Film Resistor - *Replace R3 LN1*	Antique Elec	3647-680K
S1	Part of R6	SPST	See R6	See R6
SPK1	Speaker	Speaker, Square, Ferrite Magnet, 2.6", 4 ohm	Jameco	99996
V1	Vacuum Tube	Dual Triode 12AU7	Antique Elec	T-12AU7-JJ
PWR 1	AC-DC Supply	Unregulated,12 VDC/750 mA	Jameco	2155006
Qty	Hardware			
1	Tube Socket for V1		Antique Elec	3398
2	3/4" Nylon Spacer #6 Hole		Digi-Key	492-1111-ND
2	#6-32 1-1/4" Machine Screws			
2	#6-32 Machine Nuts			
2	#6-32 Machine Lock Washers			
2	6 Lug Terminal Strip	2nd Lug Ground	Antique Elec	5354
2	#6-32 3/8" Machine Screws			
2	#6-32 Machine Nuts			
2	#6-32 Machine Lock Washers			
4	1/4" Nylon Spacers #4 hole		Digi-Key	492-1074-ND
4	#4-40 3/4" Machine Screws			
4	#4-40 Machine Nuts			
4	#4-40 Machine Lock Washers			
12	#8 5/8" Brass Wood Screws	Flat Head		
4	#8-32 1/2" Brass Machine Screws	Oval Head		
4	#8-32 Brass Machine Nuts			
2	#6 1/4" Flat Head Machine Screws	Trim thread length to avoid contact with moving stator of C2		
1	Solder Lug	Attach to bottom of C2	Antique Elec	4105
1	#6 1/4" Flat Head Machine Screws	Trim thread length to avoid contact with fixed stator of C2		
1	1/8" 2'x4' Tempered Hardboard		Home Depot	7005015
3	Nylon Spacer I.D. 3/8" x 3/8" x 1"		Home Depot	815118
Miscellaneous				
200'	#28 Enameled Magnet Wire		Antique Elec	5824
100'	#22 Hook-up Wire	Black	Jameco	36792
100'	#22 Hook-up Wire	Green	Jameco	36822
100'	#22 Hook-up Wire	Red	Jameco	36856
4'	Spaghetti Wire Tubing	1/16" Black Heat Shrink Tubing	Jameco	419127
1	3/8" Round Dowel		Home Depot	38-4EDC
1	3/4" x 3/4" Square Dowel		Home Depot	34-3HWSQED
1	1-1/2" Control Knob	Communication Type	RadioShack	274-0402
2	1" Control Knob	Communication Type	RadioShack	274-0416
	Wood Stain to Personal Taste			
	1/8" Picture Frame Backing			




L2 9 turns #16 wire tightly wound
Tap soldered to bare #16 at beginning of turn 8
1 To TS1 – 5
2 To TS1 - 3
3 To TS1 - 2
L1 2turns #22 hookup wire
4 To J4 - 1
5 To J4 - 2
B.

■ **FIGURE 10. Template of spiderweb shortwave coil.**

tube V1, turn the radio on with the volume control. Rotate the gain control fully clockwise. Use an ohmmeter to measure resistance between pins 1 and 2 of J1. After a few seconds, it should read 1,000 ohms or higher. A low reading (less than 100 ohms) suggests incorrect wiring and should be investigated.

Next, insert V1 into its socket and repeat the measurement. Now, expect a low resistance of 27-30 ohms. When satisfied with these checks, you are ready for the "smoke test." Turn the radio on and set the volume control, gain, and tuning dials to mid-position. Apply either a 12 volt battery or AC-to-DC source, and note whether the heaters in V1 glow a dull red.

After warm-up (about 30 seconds or so), you should hear static in the speaker. Rotate the gain clockwise until you hear a squealing sound, indicating that the regenerative detector has passed into full oscillation. Normally, you will operate with the gain set below this point. On especially weak signals or when full selectivity is needed, set the gain just below the point of oscillation.

Operation

In urban areas with strong AM stations, the spiderweb loop will be all the antenna needed. Best reception of distant stations will be at night using an outside antenna 25′ to 50′ in length in conjunction with an earth ground. Here again, the Internet will provide lots of advice on installing long wire antennas and earth grounds. Here's a tip when finding really weak stations.

First, use earphones rather than the speaker to eliminate distracting sounds around you. Next, rotate the gain control until the radio just breaks into oscillation. As you rotate the tuning knob, you will hear whistles that are heterodynes or beat frequencies of the radio's oscillation and radio stations. Rotate the tuning knob very slowly, and note that the beat frequency starts at a high pitch and decreases as you rotate the tuning knob. When the pitch is very low or disappears completely, you are tuned directly on the station's frequency. If you tune too far, the pitch will begin rising.

When "on frequency," reduce the gain until the detector just falls out of oscillation, and you should hear the station. It will likely be weak and fade in and out, so you will have to listen carefully to hear the station identification and get the call sign and location.

Circuit Modifications and Enhancements

My initial choice of frequency coverage was the AM broadcast band, but the radio can tune the shortwave bands by simply changing the spiderweb coil design. **Figures 10A** and **10B** show a coil designed to cover the 4-14 MHz shortwave bands, including international broadcasts and amateur radio. It requires heavier gauge wire (#16) and larger slits in the spiderweb coil. Because of the very wide tuning range, you may have difficulty tuning signals precisely.

Shortwave receivers often have a second tuning capacitor of a much smaller value in parallel with the existing one. This "band spread" capacitor provides easier and more precise tuning once the general frequency is tuned with the main tuning capacitor.

Rather than adding another variable capacitor, a varactor diode could be used with the band spread tuning accomplished by a potentiometer controlling the reverse voltage of the diode. I have not tried this yet, but see no reason why it wouldn't work.

I hope you will enjoy building and playing the retro regen radio as much as I have. Though I have not accomplished it yet myself, maybe we will succeed finally in accomplishing the goal of "Europe on one tube!" **NV**

Beyond the Arduino 3

The “Ins” of an AVR Microcontroller

Last month, we managed to get our breadboard-based AVR microcontroller project blinking at us using Atmel Studio. This month, we’re going to focus on interacting with the bare-metal microcontroller by handling inputs — both digital and analog. We’ll also pull the first three articles together by building a simple game.

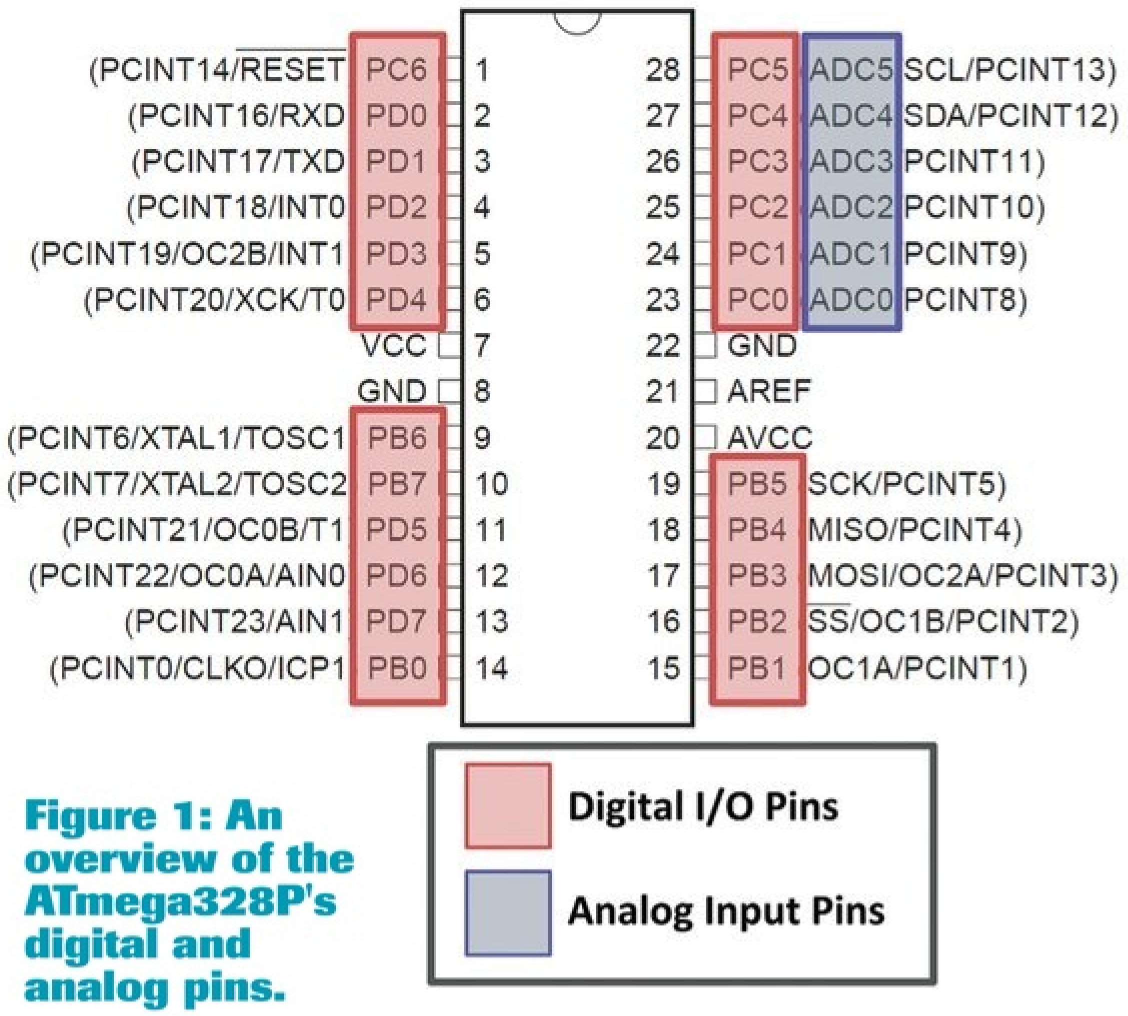


Figure 1: An overview of the ATmega328P's digital and analog pins.

Stop the Blinking Light!

I was really excited when I made my first project blink. Talking to others, they usually feel the same. (I bet you did too!) There’s something mesmerizingly satisfying about making a light turn on and off. However, it becomes a little repetitive after a while — on ... off ... on ... off ... once every second. A thousand blinks later and it’s just plain annoying — probably has you reaching for the power connector! This month, we’ll learn how we can stop that blinking light (without pulling the power) by building projects to interact more fully with the user — we’ll be exploring digital and analog inputs.

Flashback to the Arduino Uno

This series is intended to take you *beyond* the Arduino. Before we kick off with the bare-metal microcontroller, let’s quickly look back at how the Uno handles inputs so we have a common departure point.

Digital Inputs

The Arduino boards have a number of pins marked as digital only; on the Uno, these are pins 0 to 13. In addition to these 14 pins, you can use the analog pins (A0 to A5) as digital inputs.

To read a digital input from a pin, you need to:

1. Let the Arduino Uno know you want to use the pin as an input, and set the internal pull-up resistor if needed by using the *pinMode()* function.
2. Read the value from the pin, using the *digitalRead()* function.

Remember when using inputs that you must connect a pull-up or pull-down resistor, or use the internal pull-up resistor on the pin. This stops the pin from floating by pulling it to a known high/low state. A look at the ATmega328 datasheet tells us that the internal pull-up resistor ranges between 20 kΩ and 50 kΩ.

Analog Inputs

Only pins A0 to A5 can be used as analog inputs on the Arduino Uno. They return a value from 0 to 1023 (i.e., they have 10-bit precision) which represents voltage levels between 0V and a specified “voltage reference.” On the Uno, this voltage reference defaults to 5V. We’ll touch more on reference voltages later.

To read an analog input from a pin, you need to:

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/may2015_Retallack**.

By Andrew Retallack
www.crash-bang.com

Direction Pins	Input Pins	Output Pins
DDx0	PINx0	Px0
DDx1	PINx1	Px1
DDx2	PINx2	Px2
DDx3	PINx3	Px3
DDx4	PINx4	Px4
DDx5	PINx5	Px5
DDx6	PINx6	Px6
DDx7	PINx7	Px7

Table 1. Summary of macros for digital pin I/O ("x" refers to the port A/B/C/D).

1. Set the pin as an input using the *pinMode()* function.
2. Ensure that the internal pull-up resistors are not enabled.
3. Read the value from the pin using *analogRead()*.

A Quick Look at Macros

Macros are set using the *#define* statement, and are basically fragments of code that are substituted into your code at compilation. They can range from simple fixed values to full functions. For example, you could write a segment of code like this:

```
#define PERIOD 30
. . .
If (someTimer < PERIOD)
{
//do something
}
```

When the code is compiled (with the pre-processor actually doing the work), the macro *PERIOD* is substituted so the code effectively looks like this internally:

```
If (someTimer < 30)
{
//do something
}
```

The macros listed in **Table 1** are *#defined* in the same way, but you'll need to track their origin through the io.h *include* file. This *include* file makes sure that the right *#define* macros are available to you for the microcontroller that you're using.

As macros are basically code substitutions, you can make them do more than simply replace a value. Take a look at the example below; it's almost a function. See if you can figure it out:

```
#define setBit(port, bit)
( port |= (1<<bit) )
```

Macros that are of the function type can make code easier to write and read — or more challenging depending on how well they're used. However, as with any abstraction, making things easier may make them slightly less efficient.

Let's Get Digital

Reading digital inputs from Atmel Studio is pretty similar to how you're used to doing it using the Arduino IDE (Integrated Development Environment); well, the steps are the same but the syntax is different. Firstly, you need to set the direction of the pin to an input. Secondly, you need to set the internal pull-up resistor if you need it. Finally, read the value on the pin — although this is a little more complex as you need to use a mask (no, not that kind of mask!). **Figure 1** shows the available digital pins.

Setting the Direction

In the previous article, we used the Data Direction Register (DDRx) to specify the direction of a pin: input (0) or output (1). We also went into a whole lot of detail on bit shifting and bitwise operations. When I wrote my first few programs in Atmel Studio, I needed to continually refer back to the bitwise operation summary I had made before I finally managed to get it drilled into my head. If you're anything like me, take a moment to revisit the previous article — it'll make things run more smoothly from here.

To designate a pin as an input (for example, pin PB4), you'll use the following code to ensure it has a value of 0 (in other words, bit 4 in DDR B is cleared):

```
DDRB = DDRB & ~(1<<4);
```

A Digression: Some Shorthand to Make Life Easier

It doesn't take much to convince me to head off on a tangent, but it's (usually) for a good reason. I'm sure you're already finding the above code to be a bit of a "mouthful" to type each time (pardon the mixed metaphor). Thankfully, there is a way to make your code easier to enter and easier to read.

You probably know that the C language has a useful shorthand notation that makes an "x = x <*operation*>" easier to type. For example:

- `x = x + 1` can be replaced with `x += 1`
- `x = x * 7` can be replaced with `x *= 7`

The same notation can be applied to bitwise operations. So, we can simplify the setting of the Data Direction Register above by replacing *DDRB = DDRB &* with *DDRB &=* so that it reads:

```
DDRB &= ~(1<<4);
```

This is a little quicker to type, (to my eyes) read, and understand. I'm happy to take all the help I can get!

To make our code even easier to read, there is a set of macros that define the pin numbers (and therefore the shift needed on them). Remember that a macro is sort of like a cross between a function and a static variable. Refer to the sidebar, *A Quick Look at Macros* for a little more detail on these wonders. **Table 1** summarizes the I/O macros for the ATmega328P. So, after working through my not very short description of shortcuts, you can simplify the original statement:

```
DDRB = DDRB & ~(1<<4);
```

to:

```
DDRB &= ~(1<<DDB4);
```

That simplification may not seem that simple, but after

Action	In Code	Shorthand
Set PC3 to an output	DDRC = DDRC | (1<<3)	DDRC |= (1<<DDC3)
Set PC3 to an input	DDRC = DDRC & ~(1<<3)	DDRC &= ~(1<<DDC3)
Set PC3 high	PORTC = PORTC | (1<<3)	PORTC |= (1<<PC3)
Clear PC4	PORTC = PORTC & ~(1<<4)	PORTC &= ~(1<<PC4)
Toggle PC2	PORTC = PORTC ^ (1<<2)	PORTC ^= (1<<PC2)

Table 2. Examples of port manipulation in shorthand format.

using it a handful of times I got used to it and found it really helped in my coding. **Table 2** is similar to the summary I made when using Atmel Studio for the first time, and gives some common examples for us to refer back to.

Back on Track: Pulling Up

After that slight digression, let's get back to reading the digital input. We've set the DDR so that PB4 is an input. Now, let's use the internal pull-up resistor to pull the pin high. We do this by setting the relevant bit in the PORTB data register high — the same register you would use to (for example) turn an LED on. Some other microcontrollers have a dedicated register to control the pull-up resistors, but the ATmega family simply uses a combination of direction and output registers to set this.

You may have come across this behavior in older versions of the Arduino IDE in which you used the *digitalWrite(pinX, HIGH)* to enable the pull-up; versions of the Arduino IDE from 1.0.1 allowed you to set the pull-up using *pinMode(pinX, INPUT_PULLUP)*. So, to enable the pull-up (and using the shorthand), we would enter:

```
PORTB |= (1<<PB4);
```

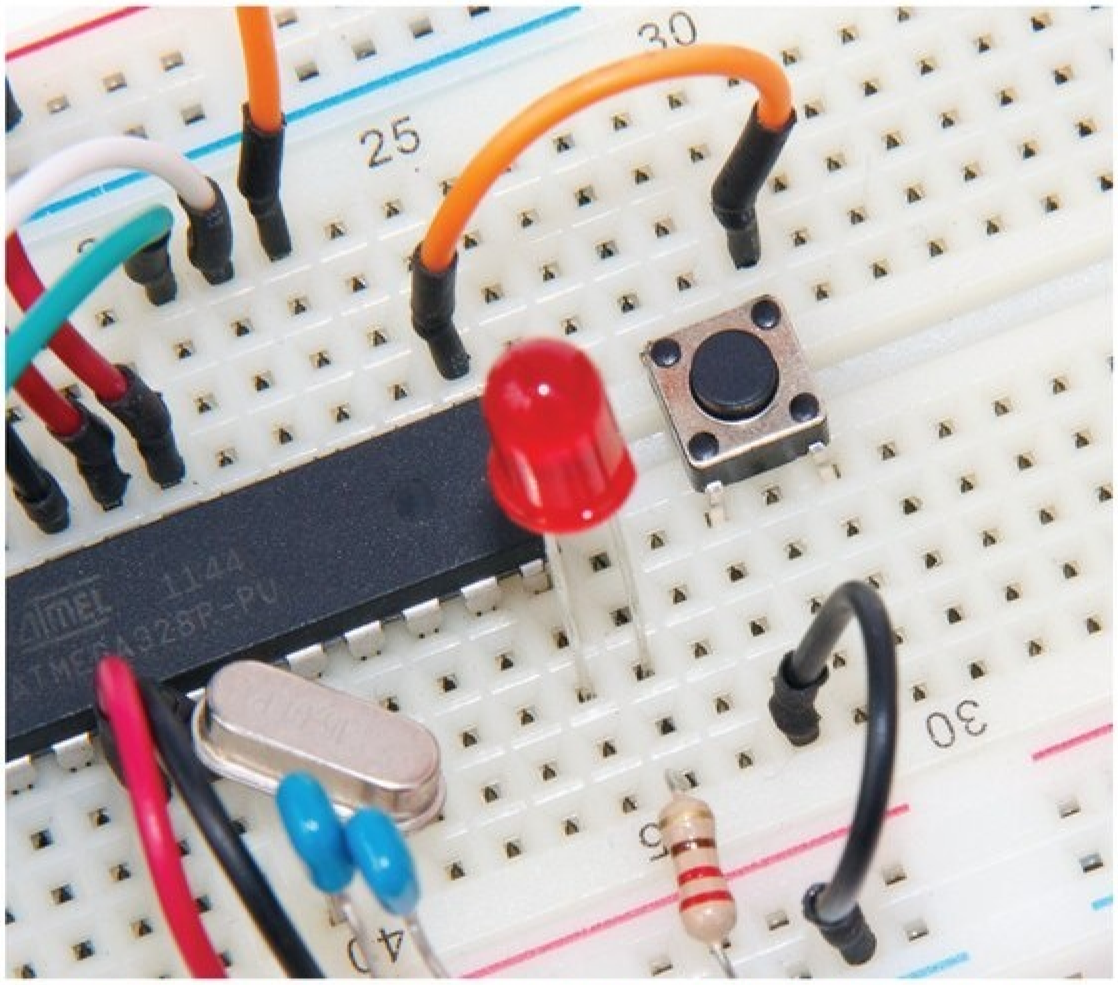


Figure 2: Half-Time project on a breadboard; use a button to control an LED.

Let's Finally Read the Input

The final step (it's taken a while to get here!) is to actually read the input. As you've guessed, this will involve another register: the input register (PINx). As with all the registers we've used so far, each input register stores the inputs for an entire port (A/B/C/D). So, the PINB register stores the input value of PB0 in bit 0, PB1 in bit 1, all the way through to PB7 in bit 7.

So far, we've focussed on *setting* individual bits, but not on *reading* them. How do we extract the value of a single bit (say, bit 4 of PINB) to see whether PB4 is high or low? In the Arduino IDE, we used *digitalRead()*; in Atmel Studio, we need to find another technique.

The technique we need is something called **bit masking**. This was a term that used to terrify me whenever I read it, and often had me thinking about running back to the Arduino IDE. I stuck it out, and now it's second nature. The term "bit masking" actually describes what it does. It masks (or hides) all the bits except for those that we're interested in. So, in this case, we'll mask out all the bits except for bit 4.

The bitwise AND operator is how we achieve this. You may remember from the previous article that the AND operator only returns a 1 if the matching bits in both expressions contain a 1. For example:

```
    0b 1 1 1 1 0 1 0 1
AND 0b 0 0 0 1 0 0 0 0
    ------------------
=   0b 0 0 0 1 0 0 0 0
```

(Only bit 4 is a 1 in both expressions)

We can use this to test whether a specific bit is high or low. If we want to see whether bit 4 of the PINB register is high, we AND (using the &) the value of the PINB register with PINB4. If the result is not a 0, then we know the pin is high. Using our shorthand, then:

```
If ((PINB & (1<<PINB4)) != 0)
{
   //Do something if pin PB4 is HIGH
}
Else
{
   //Do something if pin PB4 is LOW
}
```

That's all we need to read a digital input. The steps are the same as those we followed with an Arduino board, with the syntax differing. Remember that if you're using a pull-up resistor, the pin will normally be high and will go low when, for example, a button is pressed — reverse logic!

A Half-Time Project

Before we move onto reading analog inputs, let's pull

```
 /*
 * Beyond Arduino 3 - 1 Button-LED.c
 *
 *   Nuts & Volts - Beyond Arduino #3
 *  ------------------------------
 *   Use a push-button on PB1 to activate an
 *   LED on PB0, illustrating digital inputs
 *
 *  Author: Andrew Retallack
 *  www.crash-bang.com
 */

#include <avr/io.h>
 //Standard support for AVR I/O registers

int main(void)
{

 DDRB |= (1<<DDB0);
 //Set PB0 as output - same as pinMode(x, OUTPUT)
 DDRB &= ~(1<<DDB1);
 //Set PB1 as input - same as pinMode(x, INPUT)
 PORTB |= (1<<PORTB1);
 //Enable the Pullup resistor on PB1

  while(1)
  {

 //When reading the button value, remember the
 //pullup resistor: PB1 will be HIGH when button
 //is not pressed.  PB1 will be LOW when button
 //is pressed

   if ( (PINB & (1<<PINB1)) == 0 )
 //Check whether PB1 is LOW, using bit masking
   {
 //PB1 is LOW:  Therefore button IS pressed
          PORTB |= (1<<PORTB0);    //Set LED on
   }

   else
   {
 //PB1 is not LOW:  Therefore button NOT pressed
       PORTB &= ~(1<<PORTB0);    //Set LED off
   }

 }
}
```

Listing 1. The Guessing Game project.

The projects here are based on the build from "Beyond the Arduino — Part 1" in the March edition of *Nuts & Volts*. The following additional components are needed for the final project:

D1	Green LED 20 mA
D2	Red LED 20 mA
R1, R3	330 ohm resistor, 0.25W
RV1	Potentiometer
SW1, SW2	Momentary pushbutton switches

this together with a quick example. Wire up a pushbutton and an LED (including the limiting resistor) to your breadboard microcontroller as in **Figure 2**. Then, create a new Atmel Studio project and download **Listing 1** from the article link.

Compile and upload the code. You should notice that the LED turns on each time you press the pushbutton — a simple (and not particularly useful) example to read a digital input.

I'd like to share a quick tip about programming your board. As I use my programmer for multiple projects, I often need to connect and disconnect it — which can be a slow process if I'm working out which programmer pin to connect to which microcontroller pin. I make this easier by building a "header" on the breadboard with a pin configuration that matches the programmer. Now when I connect the programmer, I just need to make sure it is oriented the correct way, and don't have to worry about individual pin connections. **Figure 3** highlights this on the breadboard.

Hold On, What's the Benefit?

If you aren't catching your breath right now, then you're a stronger person than most people I know when they first tackle the transition to Atmel Studio. Personally, I struggled to see how I was benefitting from complicating my life. I had thrown out perfectly good and easily understandable functions, and was now writing my programs in a bizarre form of barely understandable cryptic code.

There are a few reasons why working directly with registers can be better:

- Less abstraction means a smaller code footprint and slightly improved performance.
- By using bitwise operations, you can control an entire set of pins in a single statement rather than multiple Arduino statements.
- Probably a few other trivial reasons ...

Not convinced? Well, to be honest, you won't really see much benefit from what we've done so far. The key

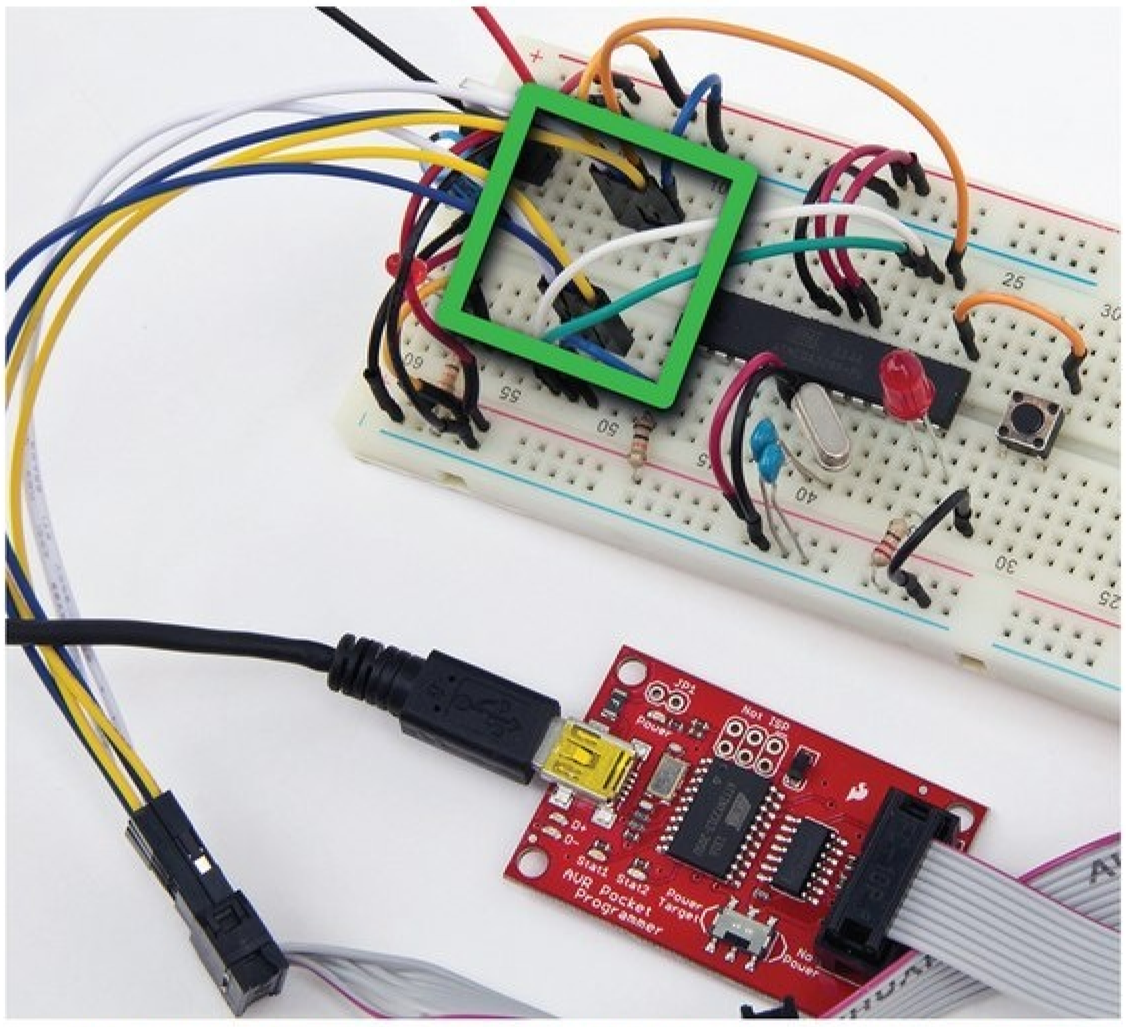


Figure 3: Make programming easier with a set of connectors to match your programmer's.

ADC Successive Approximation

The ATmega328 ADC uses a method called successive approximation to convert an analog signal into a digital value, meaning that it is equipped with an SAR (Successive Approximation Register) ADC. Don't be put off by the mouthful of mnemonics as it's actually quite a simple concept.

The SAR-ADC works by generating a voltage using a DAC (Digital-to-Analog Converter). I know, it sounds rather back to front that the ADC in reading a voltage, in fact, generates a voltage. However, it uses it in a pretty smart way, utilizing a binary search (for those of you that are from a software background).

An example is probably the easiest way to demonstrate this. I won't go the full 10 bits of the ADC's resolution as I know you have a life outside of this sidebar, so let's keep it to six bits.

Let's say that our AREF (remember, this is our voltage reference) is 5V, and that the voltage on pin PC0 is 1.68V. The ADC finds the voltage by:

1. Generating a voltage on the DAC.
2. Comparing the DAC voltage to the voltage on the analog pin.
3. If the DAC voltage is higher, set the current bit in the SAR register to 0, and subtract half the difference between the current and previous DAC value.
4. If the DAC voltage is lower, set the current bit in the SAR register to 1, and add half the difference between the current and previous DAC value.

Figure A shows a graph of the approximation process using a six-bit ADC, resulting in a SAR register value of 010101. This is 21 decimal (the maximum value for a six-bit ADC is 63), resulting in a calculated voltage of:

21 / 63 * 5V = 1.667V

This is pretty close to the actual voltage on pin PC0.

As you can see, there is a fair amount of work involved in taking an ADC reading – which explains why it is not an instantaneous process, and therefore why we need to wait in a loop until the conversion is complete.

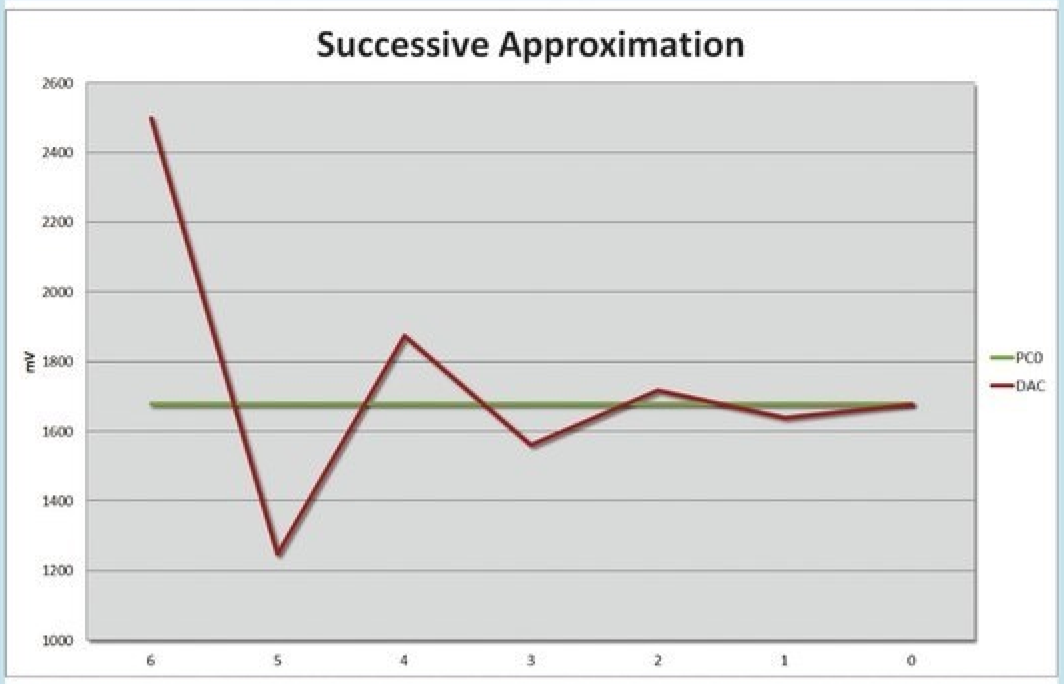


Figure A: A graph illustrating successive approximation on an ADC.

phrase there is "so far." You won't see much benefit when performing simple digital I/O, but you will see great benefits as we continue on our journey. Because digital I/O is so straightforward, it is the perfect place to cut your teeth on the concepts of registers, bitwise operations, bit masking, and so on.

If you're ready to go on and don't feel like I've been cheating you by making things complex for no good reason, then fasten your safety belt as we move onto reading analog values.

Advancing with Analog

Grab yourself a cup of coffee; it's going to get fun in this section! You likely already know that microcontrollers use an ADC (Analog-to-Digital Converter) to read analog inputs and convert them to digital values that they can work with. The sidebar, *ADC Successive Approximation* gives you a quick look into how the ATmega ADC actually does this.

Before we get into the "how," I've listed a few features of the ADC that we can access now that we've moved away from the Arduino IDE (I'm working hard to sell the benefits!):

Conversion Modes: Single, Continuous, and Triggered

In the Arduino IDE, you worked in single mode; in other words, the ADC performed a conversion each time it was instructed to with the *analogRead()* function. Working directly with the ADC, you can now choose to work in continuous mode (or free-running mode as the datasheet refers to it) in which a new conversion starts as soon as the previous one has completed. Alternatively, you can specify that the conversion is triggered by an interrupt — either an internal interrupt such as a timer, or an external one such as an interrupt on an I/O pin.

Conversions can Cause Interrupts

Using registers, you're able to configure the ADC to trigger an interrupt when a conversion has been completed. As we move through this series, we'll start using various interrupts increasingly, and the ADC interrupt can be a useful one — resulting in simpler code and saved resources.

Speed vs. Accuracy

If the standard 10-bit ADC accuracy is not needed, you are able to trade accuracy for speed. The Arduino IDE's *analogRead()* function can perform approximately 10,000 conversions a second. Working with the ADC directly at the same level of accuracy, you can achieve about 13,000 conversions a second. From there, you can increase the conversion frequency further, for a reduction in accuracy — perhaps this is an experiment to include in a later article.

I won't cover all these features this time, but it is useful to know they exist. Finally, before we get into the details, I recommend you read section 24 of the ATmega datasheet (**www.atmel.com/devices/ATMEGA328P.aspx**). It may seem like a foreign language to you, but the more you read these seemingly endless documents, the more you'll start to see their usefulness.

The 0001, 0010, and 0011 of Analog

Let's get into the 1, 2, 3 of analog by working through the steps needed to read an analog value from pin PC2. Why PC2? No real reason, other than it's an analog pin.

Analog Pins and Channels

There are six analog pins on the ATmega328P-PU: PC0 through PC5. These can be identified as analog by the labels ADC0 to ADC5 on the pin configuration diagram (refer again to **Figure 1**). If you read the datasheet, you probably picked up that there are, in fact, nine analog channels: 0 to 8. It originally took me a while to find the missing channels, but looking through the sheet I found that channels 6 and 7 are not available on the 28-pin PDIP package that we use on our breadboard. They are only on the 32-pin packages (MLF and TQFP). Channel 8 *is* available to use, and is connected to the chip's internal temperature sensor. However, this sensor only has an accuracy of ±10°C, so it has limited uses.

Typical Process Flow

Let's go through the process flow for a single conversion triggered in code; in other words, we will instruct the code to take a single reading without relying on the continuous or triggered modes I touched on earlier. I've mapped out the process flow in **Figure 4** for reference as we go.

Firstly, we will initialize the ADC before entering the main *while(1)* loop in the code. This is normally a once-off configuration step, and is something that we'll do in the future with other peripherals on the microcontroller.

Secondly, we'll initiate the conversion and then wait for it to complete — either in a loop or by using an interrupt. In this example, we'll keep things straightforward and use a loop to wait for the completion.

Finally, we'll read and interpret the value, and then do something with it.

Getting a Handle on Registers

To do all this initializing, conversion, and reading, we'll need to use a number of registers. These are used in the same way as the Data Direction, Digital Output, and Digital Input registers we've used so far, but get a little more complex. Each of the bits in the registers we've used already relate directly to a pin on the controller. The registers we'll be using here don't relate to pins, but instead either control settings or allow you to read values. Here's where the dreaded datasheet comes in handy!

Figure 5 shows an excerpt from the datasheet. The table at the top shows each bit in the register, a cryptic abbreviation of what the bit does, whether it is read/write, and finally the initial value at power-on. Below the table is a more detailed description of each bit or range of bits. As we work through the steps, refer to the relevant registers in the datasheet to see how they are used.

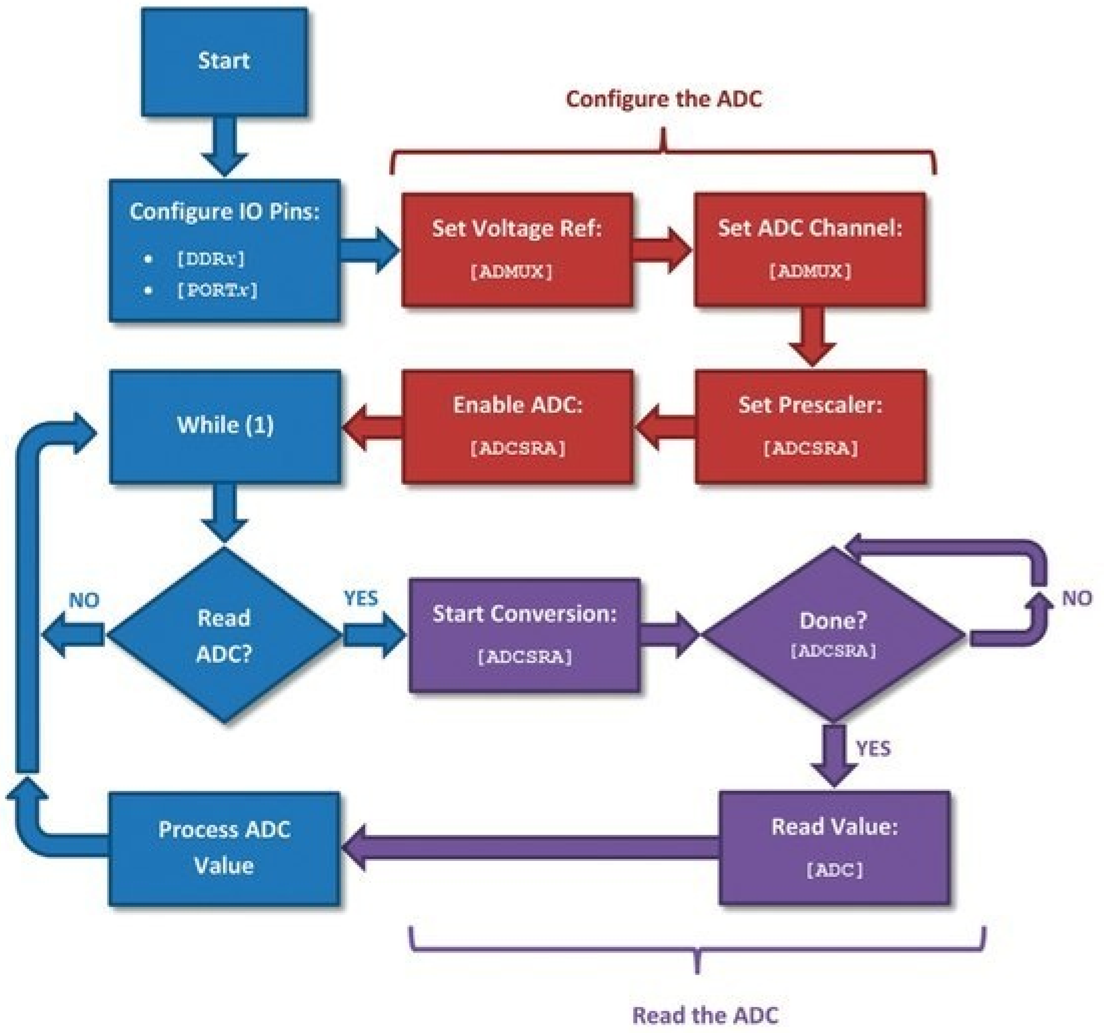
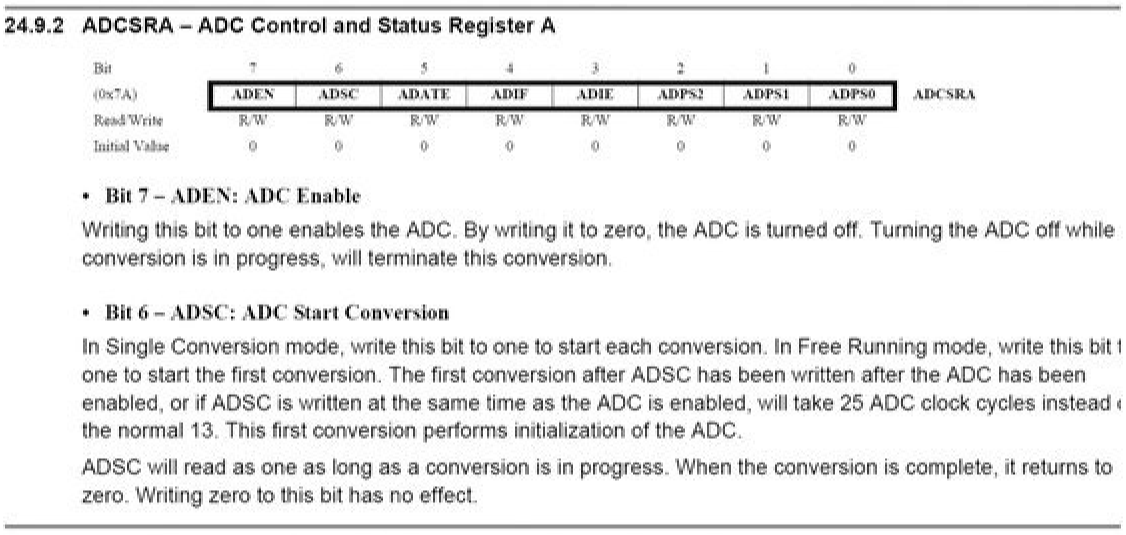


Figure 4: Overview of the ADC process flow.



24.9.2 ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – ADEN: ADC Enable**

Writing this bit to one enables the ADC. By writing it to zero, the ADC is turned off. Turning the ADC off while conversion is in progress, will terminate this conversion.

- **Bit 6 – ADSC: ADC Start Conversion**

In Single Conversion mode, write this bit to one to start each conversion. In Free Running mode, write this bit t one to start the first conversion. The first conversion after ADSC has been written after the ADC has been enabled, or if ADSC is written at the same time as the ADC is enabled, will take 25 ADC clock cycles instead the normal 13. This first conversion performs initialization of the ADC.

ADSC will read as one as long as a conversion is in progress. When the conversion is complete, it returns to zero. Writing zero to this bit has no effect.

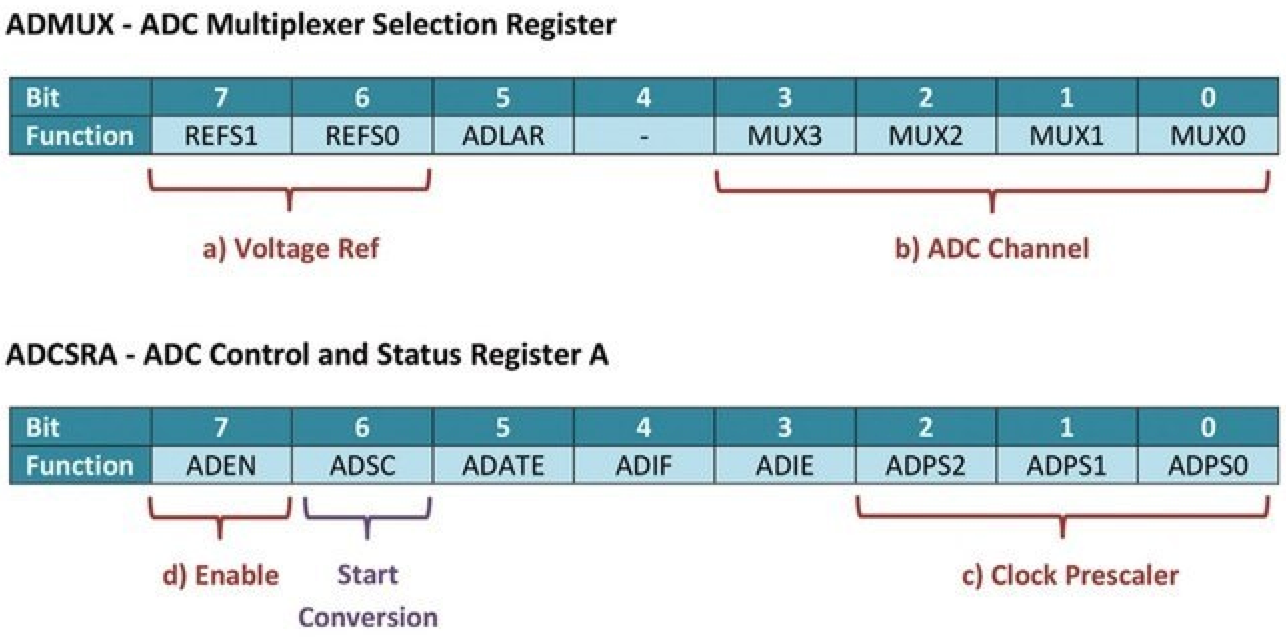
Figure 5: Datasheet extract showing a typical register description.

Using Include Files

Often, when writing a set of related functions, it is helpful to group them into separate files. For example, we could create an ADC.c and an ADC.h file to contain all the ADC-related functionality and variables. This both reduces the complexity of the main program file, and allows you to re-use all those ADC functions in other programs. We'll work through the details of creating these files in a later article, so if this example seems a little unstructured with all the code in a single "c" file, bear in mind that there are better ways to design your project.

Step 1: Initialize the ADC

This is the most complicated step in the process, so I usually wrap it up in a separate function that I call from my main routine. Once the ADC is configured, it is quite



ADMUX - ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0
Function	REFS1	REFS0	ADLAR	-	MUX3	MUX2	MUX1	MUX0

ADCSRA - ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0
Function	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0

Figure 6: Overview of registers and bits needed to initialize the ADC.

straightforward to take a reading. As you work through the configuration, you'll hopefully start to see how the Arduino IDE hid the complexity, and by necessity also hid the flexibility. The steps detailed next are highlighted in red in the process flow in **Figure 4**. Additionally, **Figure 6** summarizes the registers and the bits that we use when initializing the ADC.

A) Set the Voltage Reference

The voltage reference is the highest voltage that you expect to measure on the input pin. The reason you would want to alter this is that it affects the resolution of the readings you take. It's probably easiest to explain through an example:

The 10-bit ADC returns a value from 0 to 1023 — proportional to the voltage that has been read on the input pin. If the reference voltage is 5V, then the range from 0 to 1023 represents a voltage from 0V to 5V; in other words, each unit from 0 to 1023 represents (5V / 1023) = 4.89 mV. If you specify a reference voltage of 1.1V, then each unit from 0 to 1023 represents (1.1V / 1023) = 1 mV; a much higher resolution.

You can probably see that it makes sense to use a voltage reference that is as close as possible to (but larger than) the maximum voltage you expect to measure, as you have a greater resolution and accuracy.

The ADC allows three types of reference voltages to be set: the voltage the ATmega is powered at (i.e., VCC); the voltage present on the AREF pin; and an internal 1.1V reference. For this example, let's use AREF, which on our breadboard is connected to the 5V power rail (the AREF is pin 21 on the ATmega328P).

To set the voltage reference, we need to set bits 6 and 7 of the ADMUX register (the ADC Multiplexer Selection Register). Don't worry if the naming sounds overcomplicated and pretty meaningless — imagine if you had to name all these registers! The important thing is what they do. So, pull out the datasheet and go to the "Register Description" section of Chapter 24. See if this makes sense to you:

```
ADMUX &= ~( (1<<REFS0) | (1<<REFS1) );
```

B) Set the ADC Channel

Remember that the ADC has eight channels (although 6 and 7 are not available on our physical version of the ATmega328). The channel numbers nicely tie up to the naming of the pins: ADC0 is channel 0; ADC5 is channel 5. We said we'd work with pin PC2, which is ADC2. Therefore, we set the channel to 2. Bits 3 to 0 specify this, so the code would be:

```
ADMUX |= 2;
```

We use the OR (|) operation to set the bit, as we don't want to change any other bits in the register that may have already been set.

C) Set the Clock Prescaler

A prescaler was something else that terrified me, and as a result my brain shut down whenever I read the term. In reality, it is a very simple concept: A prescaler slows down your system clock to a speed that the peripheral can cope with. The poor ADC has no hope of keeping up with the 16 MHz clock that the microcontroller is running at, so we slow it down by dividing it by a factor. The most we can divide it by is 128, so the ADC works at a speed of (16 MHz / 128) = 125 kHz. Far more reasonable!

The datasheet recommends that we run the ADC between 50 kHz and 200 kHz for maximum accuracy, so 125 kHz is perfect. Let's set the prescaler to 128 then; bits 2 to 0 of register ADCSRA (ADC Control and Status Register):

```
ADCSRA = (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0);
```

D) Enable the ADC

The last step is to enable the ADC. We've set the voltage reference, set the channel, and set the speed (using the prescaler). Now, we enable the ADC so it's ready for action by setting the ADEN bit of the ADCSRA register to a 1:

```
ADCSRA |= (1<<ADEN);
```

Step 2: Take a Reading

Now that we're all set up, we're ready to start reading values off the ADC. As mentioned earlier, we'll restrict this article to performing simple single conversions. All the work setting the ADC up makes the actual conversion a pretty simple thing. We need to tell the ADC to start a conversion, wait until it has finished, and then take the reading. Refer back to **Figure 4** and look at the process highlighted in purple.

A) Kick-off with the Conversion

Starting with an ADC conversion is straightforward.

Simply write a 1 to bit 6 of the ADCSRA register. Bit 6 is named ADSC (ADC Start Conversion), so we enter:

```
ADCSRA |= (1<<ADSC);
```

B) Wait for the Conversion

In our "human" time, the conversion happens ridiculously fast: about 13 clock cycles, or running using a 16 MHz crystal — about 0.0000008125 seconds! However, in microcontroller land, this is a fair amount of time, and we need to wait for the conversion to complete.

The datasheet notes that the ADSC bit (the one we used to initiate the conversion) will remain at a value of 1 until the conversion is complete. Once complete, the ADSC bit switches to a 0. That makes it pretty easy for us. All we need to do is hang around until the ADSC bit goes low. I'm sure that you can figure that out in code. Here's how I do it using a *while* loop and. of course, bit masking (because we're now reading a value):

```
while (ADCSRA & (1<<ADSC));
```

C) The Final Play: Read the Value

Finally, a simple register name – this one is simply called ADC. The ADC register (ADC Result Register) returns a 10-bit value from 0 to 1023. You'll need to make sure that the variable you're reading the value into is large enough to handle 10 bits. Otherwise, you'll have the value wrapping around back through 0. Something like an *unsigned long* or *uint16_t* will do the trick:

```
unsigned long ADCValue;
. . .
ADCValue = ADC;
```

Very straightforward! You may have spotted something strange here, though. The ATmega328P is an eight-bit microcontroller, meaning that all its registers are eight bits (from 7 to 0). How are we then reading in a 10-bit value? The ADC result is actually stored in two registers: ADCL and ADCH (low byte and high byte registers). Thankfully, these can be read through a single register — the ADC register that we used above. If you choose to read the underlying registers, read the ADCL first, then the ADCH, and finally combine them into a single value by shifting the ADCH eight bits. This adds extra work we don't need to be taking on!

Step 3: Do Something with the Reading

So, we have a reading between 0 and 1023. What now? Pretty much the same as you would have done in the Arduino IDE: Either act on the value, or convert the value to a voltage. It's really only useful converting the value to a voltage if you need to report a specific voltage; for example, if you're building a simple voltmeter project. Otherwise, it's normally enough to work with the value between 0 and 1023 itself.

As a quick reminder, we convert the value to a voltage using the formula:

Voltage = (ADCValue / 1023) * VoltageReference

Let's Get Physical

Hopefully, you've found that we've covered some new ground this month. I'm sure you didn't read through these pages purely for the theory behind the operation, so let's get physical and build a little project. I thought that a simple game would be a good way to pull together all that we've done over the past three articles.

We'll build an analog guessing game. Player 1 turns a potentiometer and then presses a button to set a value. Player 2 needs to try to guess the value the first player set. Player 2 turns the potentiometer, and then presses a second button to "guess." A red LED will light if the guess is too high; a green LED will light if the guess is too low. When the player guesses close enough to the value, then the LEDs flash.

The goal, of course, is to guess the value in as few turns as possible. This probably won't keep you entertained for hours (although my son and I played a good number of rounds), but does hopefully illustrate the concepts we've discussed so far. **Figure 7** gives a quick overview of the LED indications through the game play.

The schematic in **Figure 8** shows the connections needed, excluding the detail of the power supply and programming connections in order to keep the diagram focussed. **Figure 9** shows how the schematic translates into the final project on the breadboard.

Enter, compile, and upload the code in **Listing 2** that is available at the article link (it is not shown here due to space restrictions).

I won't go through it in detail here, as it's pretty well commented and only includes the concepts we've covered so far. It makes a good exercise for you to work through to consolidate the theory so far.

If you would like to discuss the project in more detail, then hop over to the *Nuts & Volts* Forum (**http://forum.nutsvolts.com**) – it's a great place for readers to connect with each other and authors to discuss the theory and projects in articles.

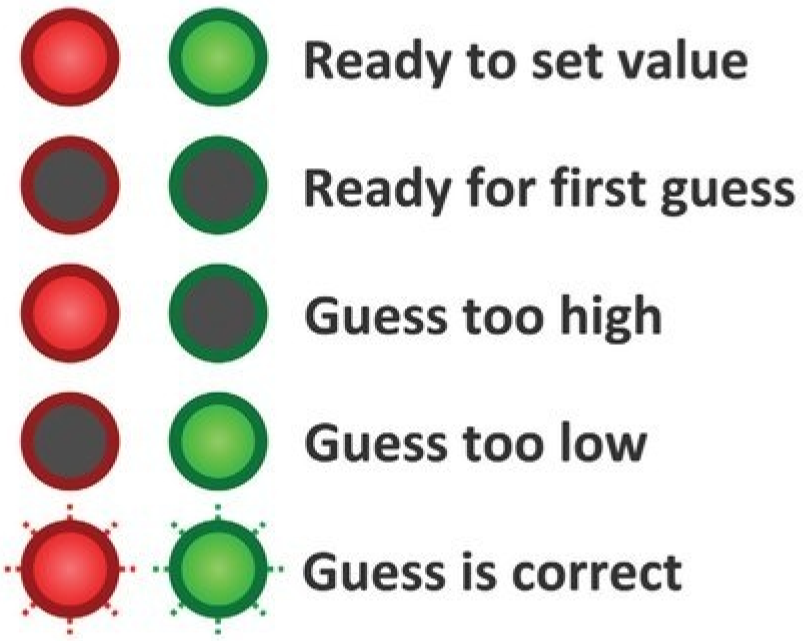


Figure 7: Guessing Game Project: Understanding the LEDs.

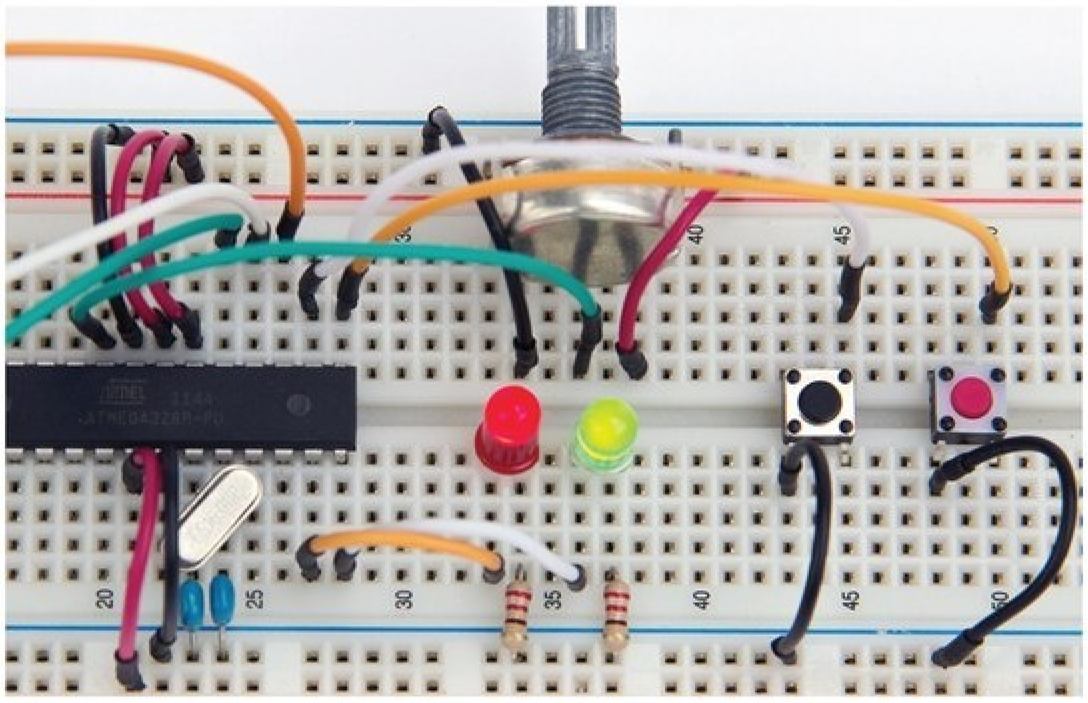


Figure 9: Guessing Game project laid out on the breadboard.

The Arduino IDE

We discussed a few IDE options in the previous article, but I wanted to elaborate on Atmel Studio vs. the Arduino IDE.

In this series, I use the term "Arduino IDE" to refer to the combination of the development environment and the built-in commands and libraries that are an integral part of it. In other words, the entire package that you download and install. I am assuming that you're using the environment as it was intended — relying on the built-in functions to make your life easier.

You can, if you choose, ignore the built-in functions in the Arduino IDE and work directly with the registers that we use in this series from within the IDE. For example, you are able to use the Data Direction and Input registers. I don't recommend this though, and suggest rather that you move to a more fully-featured IDE. The Arduino IDE is the best of its kind for achieving what it sets out to achieve, but once you discard the built-in functions that are core to what it is, I personally don't feel that it is the best environment to work in.

What's Next?

In addition to covering digital and analog inputs in this article, we've also spent some time getting comfortable using registers. Microcontroller programming is all about registers, so it's good to get comfortable with them!

From here on in, we can get into more "juicy" topics now that the basics are covered. Next month, we'll dive into serial communications, so that we can get our microcontroller projects talking to our PC. You'll know from your Arduino days that serial comms are a very useful way to both interact with and debug your projects.

We've had some great feedback. Please keep your comments and suggestions coming and, of course, share any projects you create based on what we've covered so far. Let's not restrict communication to that of the "serial" variety! I look forward to connecting again next time. **NV**

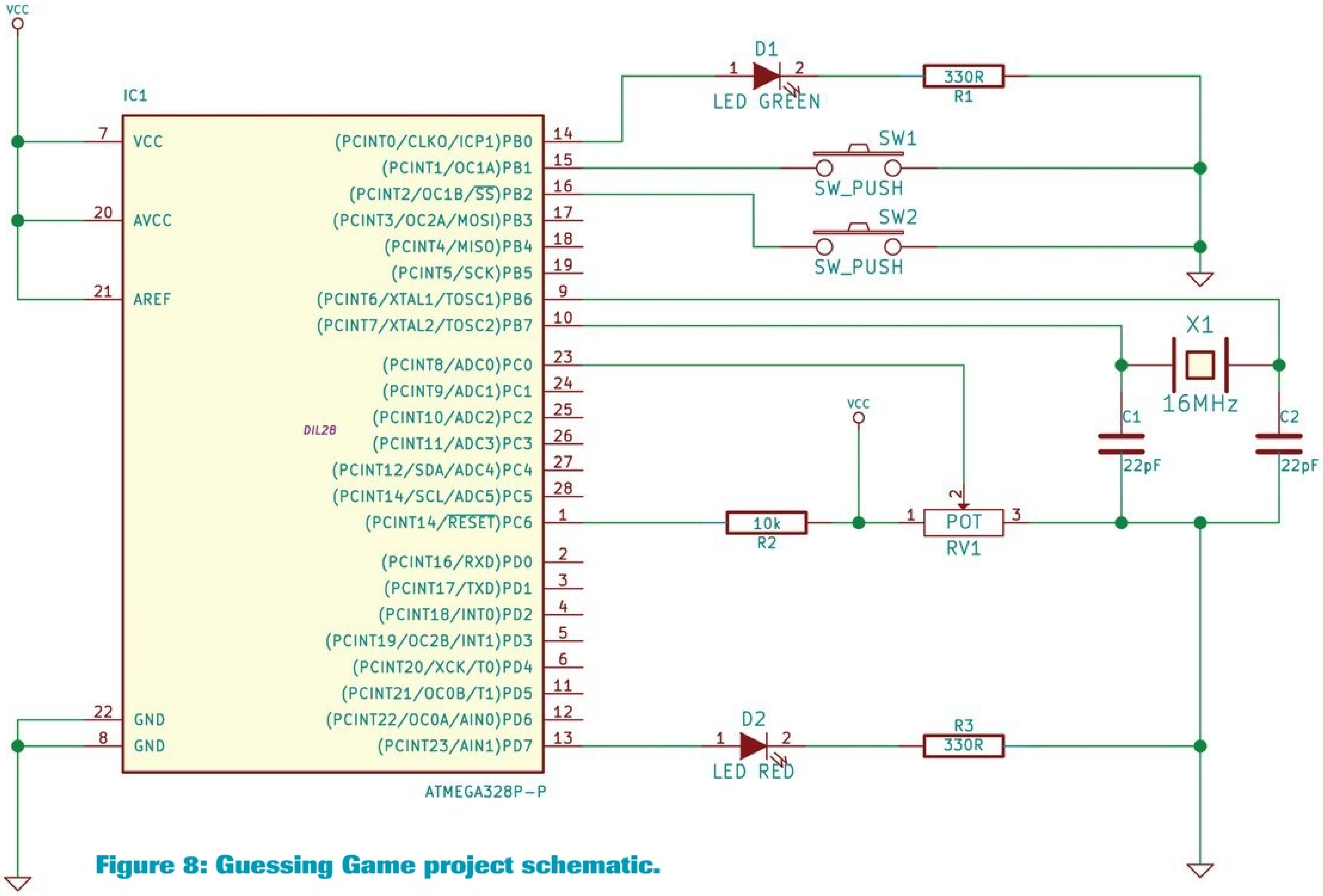
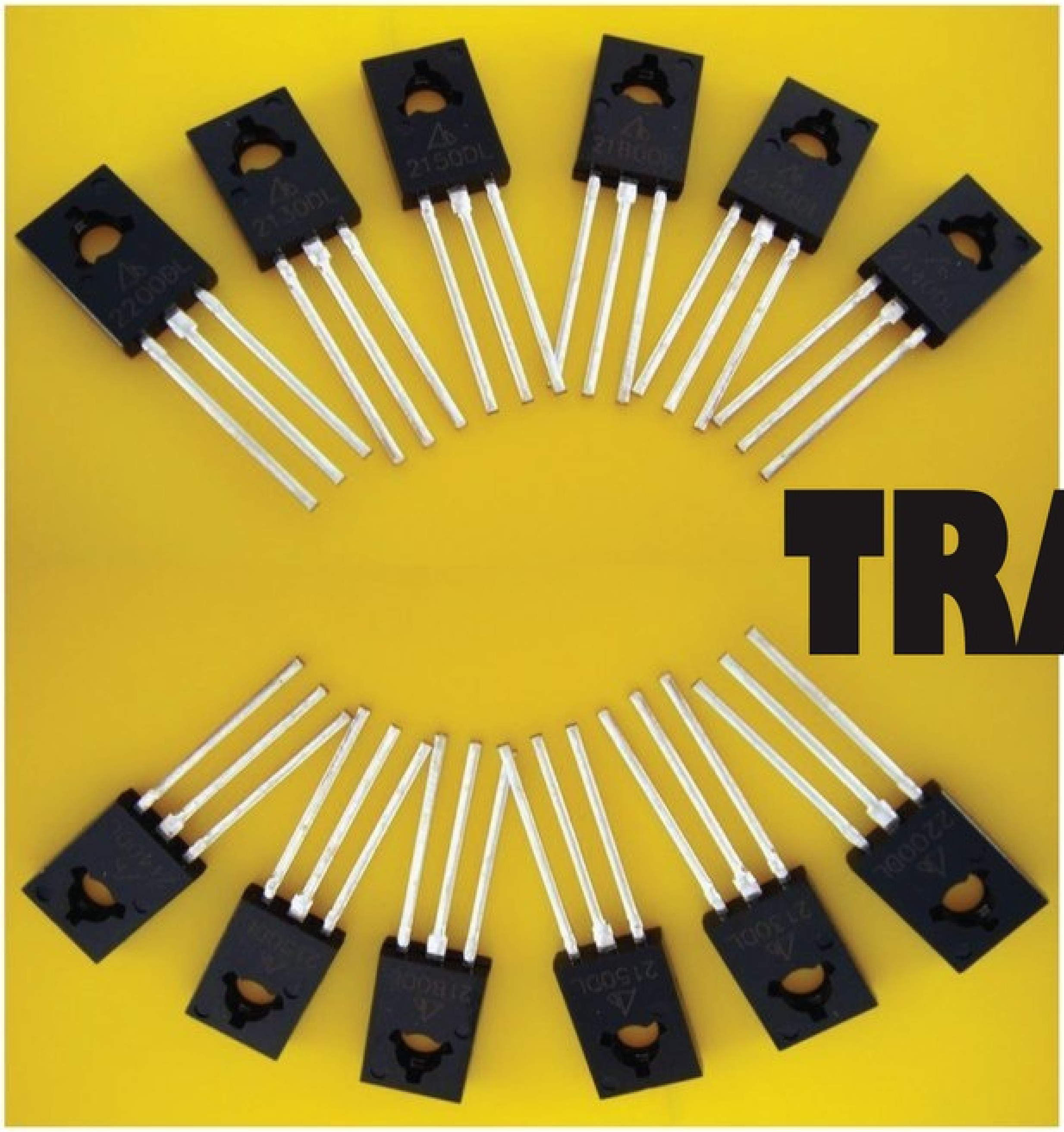


Figure 8: Guessing Game project schematic.



HOW TO USE A TRANSISTOR AS A SWITCH

By Roger D. Secura

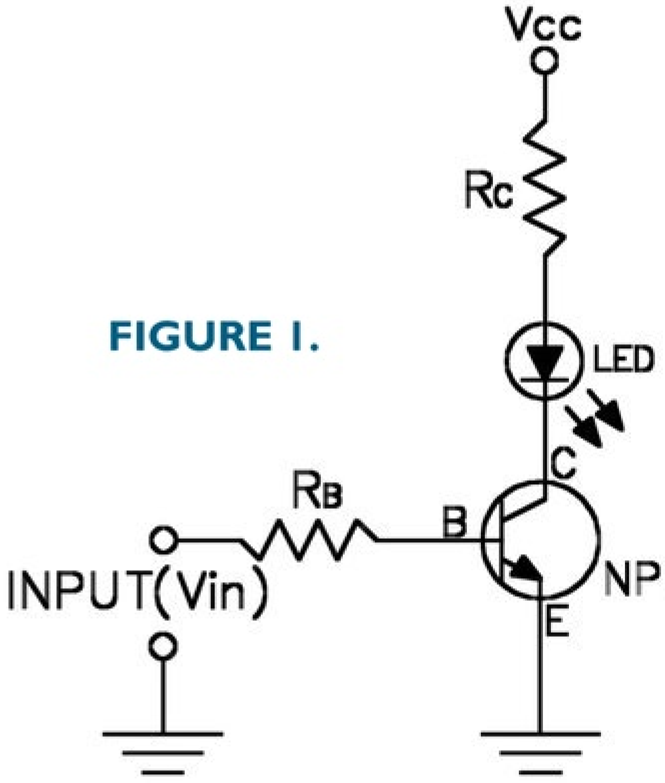
Sometimes understanding how the simplest of electronic circuits work can be intimidating and/or confusing for the student, hobbyist, or novice. If I remember correctly (back in the 17th century), half of the people in my electronics class dropped out after the first semester. Throw in a course on calculus and you'll witness a raging stampede for the exit door. Who do we blame for this abysmal dropout rate — students, teachers, or curriculum? Andrew Carnegie was once asked, "What's more important: labor, capital, or brains?" His reply was, "What's the most important leg on a three-legged stool?"

Unfortunately, the answer to the blame question is beyond the scope of this article. What should be important is presenting the subject of electronics in such a way that makes it easy for anyone to learn. With that in mind, let's take a look at one of those simple and ubiquitous circuits: the transistor switch. Don't worry, there's nothing more challenging here than multiplication, division, addition, and subtraction. So, let's begin!

A TRANSISTOR AS A SWITCH

Look at **Figure 1**. It shows a typical general-purpose (NPN) transistor/LED circuit. If you hook up +5 volts to Vcc in this circuit and pulse the input terminal (Vin) with +5V, 0V, +5V, etc., the LED will flash on and off accordingly. Of course, in order to make this circuit function properly, you have to calculate the correct resistor values for RC and RB. How do you do that? Well, keep reading.

FIGURE 1.



Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/may2015_Secura**.

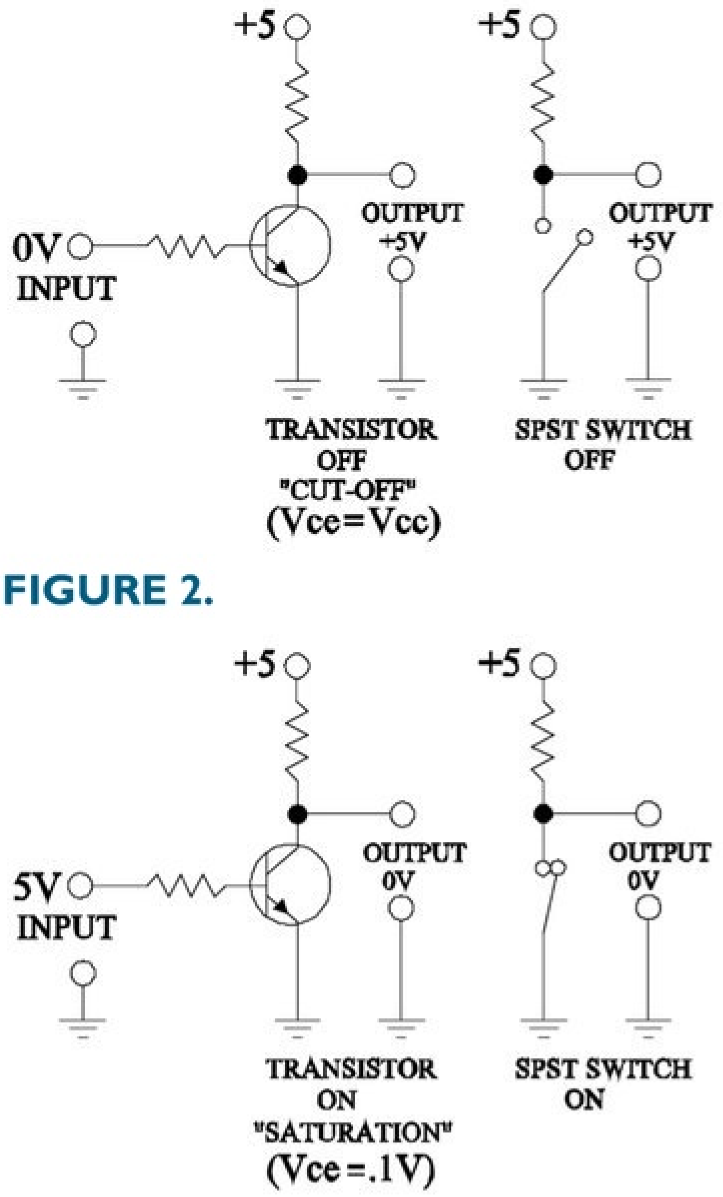


FIGURE 2.

Before we start, it should be noted that we'll be using the transistor as a simple switch and not as an amplifier. Remember, transistors can operate either as an amplifier or a switch. If you're using a transistor to amplify a signal, the transistor is said to be operating in the "active" or "linear" region.

Without going too deep into transistor theory, the active region entails a range of operating points (Google "transistor characteristic curves") at which the transistor will amplify a signal without distortion.

On the other hand, when the transistor is used as a switch, it is operating in what is called the "digital" mode (on/off). In this state of operation, the transistor does not "amplify" the input voltage (Vin) in any way. To use a transistor as a switch, all you have to do is increase the current at the base terminal to a certain level, and the transistor will go into a state commonly known as "saturation." This is a state (mode of operation) where no matter how much additional current is pumped into the base terminal of the transistor, the collector current will not increase any further.

Once a transistor is in saturation mode, it acts just like a closed SPST mechanical switch (see **Figure 2**). In turn, when the transistor is turned off (no base current), it goes into "cut-off" mode (fully off). Simply put, the transistor is either on or off — amplification is immaterial.

Okay, now that you know the difference between a transistor amplifier and a switch, let's use the transistor as a switch in order to flash an LED on and off.

CHECK THE DATASHEET

The first step is to Google the datasheets for both the LED and the transistor. You'll notice on the LED datasheet a listing for the maximum forward current (I_F). Most of the popular 5 mm diameter through-hole LEDs have a maximum current rating somewhere around 20 mA.

Once the maximum LED rating is established, what do we do with that information? Well, it means we need to reduce the max rating of 20 mA to a safe current level so the LED isn't destroyed. A good starting point is somewhere between 5 and 15 mA — depending on how bright of an LED you need. Let's agree here to set the maximum current ($I_{C(MAX)}$) flowing through our LED to 15 mA. Now, go ahead and use Ohm's Law to calculate the value for the collector resistor (Rc). The formula is listed below; assume we have +5V as our power supply (Vcc) and ($I_{C(MAX)}$) = 15 mA.

$$R_C = \frac{V_{CC}}{I_{C(MAX)}} = \frac{5}{.015}$$

Did you calculate a value of 333.33 ohms for RC? You are correct! Okay, stop right there — we have a problem! The formula above is missing a couple of very important electrical parameters. What's missing is the fact that both the LED and transistor — when turned on — have a voltage drop across their terminals and this must be accounted for in the formula.

A general-purpose transistor will drop about .1 to .3 volts across the collector/emitter terminals ($V_{CE(sat)}$; see datasheet) when in saturation mode (fully on). Once a transistor saturates, the collector current reaches a level or plateau where any additional increase in the base current will not cause a further increase in collector current. In "theory," at this point the collector/emitter voltage drop ($V_{CE(sat)}$) should be zero if the transistor was working like an SPST mechanical switch.

Remember, a mechanical switch has no voltage drop when flipped to the on state because there's no resistance between the contacts. On the other hand, transistors have a small

Switching Options

The transistor switch can be very useful in a variety of circuits. Unfortunately, in some cases, it's not the best solution. For example, if you're building an H-bridge circuit to control the motors on a battery operated robot, you wouldn't use four transistor switches because of the large .7 voltage drop across each of the base/emitter junctions. In this case, a power MOSFET switch would be a better choice — considering the on resistance (RDS) across the drain and source terminals (i.e., voltage drop) of a MOSFET is much lower at saturation than a BJT transistor.

Figure A shows just a few other optional switching devices you can use when a simple transistor switch or MOSFET is not applicable.

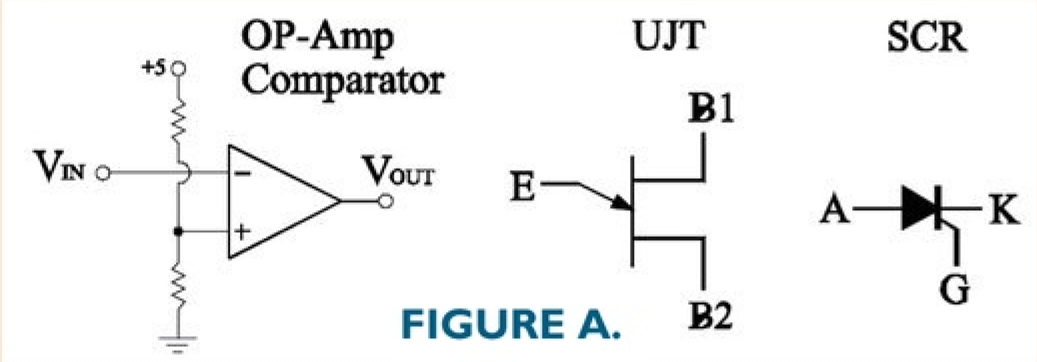


FIGURE A.

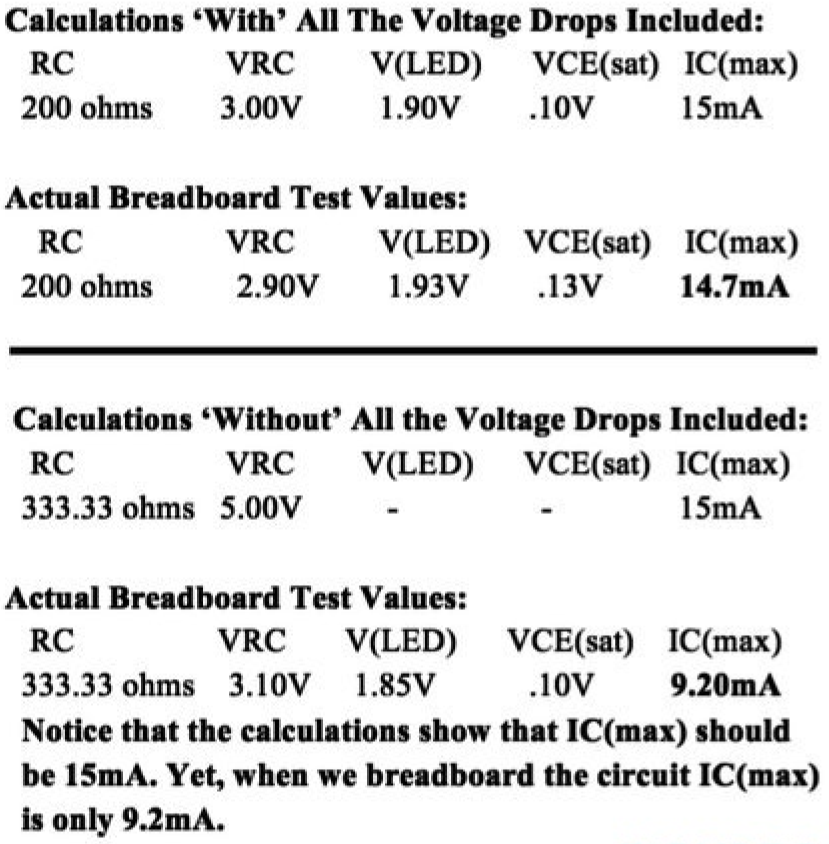
IMPORTANT POINTS

1. The circuit designer (you) determines what the correct transistor collector current ($I_{C(sat)}$) should be by looking at the LED/transistor datasheets and verifying that the current going through the transistor/LED circuit is below the maximum ratings for both devices.

In other words, the saturation current ($I_{c(sat)}$) flowing through a transistor switch is not determined by the transistor's internal electrical parameters, but rather by the external components (resistor/LED) employed by the circuit designer.

2. *B*eta (DC gain) as listed in the datasheet has no meaning when a transistor is used as a switch (saturation/cut-off). Only amplifier designers care about the various levels of collector current (gain) in between saturation and cut-off. In other words, any level of collector current in between the two operating states of "saturation" and "cut-off" (i.e., active region) is not important to the functioning of a transistor switch circuit.

3. "Saturation" in a transistor switch circuit is achieved when the voltage across the collector/ emitter ($V_{CE(sat)}$) is less than or equal to .1 to .3 volts - depending on the type of transistor. At that voltage point, the transistor appears to act like a simple SPST mechanical switch that has been closed (On).



Calculations 'With' All The Voltage Drops Included:

RC	VRC	V(LED)	VCE(sat)	IC(max)
200 ohms	3.00V	1.90V	.10V	15mA

Actual Breadboard Test Values:

RC	VRC	V(LED)	VCE(sat)	IC(max)
200 ohms	2.90V	1.93V	.13V	**14.7mA**

Calculations 'Without' All the Voltage Drops Included:

RC	VRC	V(LED)	VCE(sat)	IC(max)
333.33 ohms	5.00V	-	-	15mA

Actual Breadboard Test Values:

RC	VRC	V(LED)	VCE(sat)	IC(max)
333.33 ohms	3.10V	1.85V	.10V	**9.20mA**

Notice that the calculations show that IC(max) should be 15mA. Yet, when we breadboard the circuit IC(max) is only 9.2mA.

FIGURE 3.

amount of resistance across the collector/emitter terminals (R_{CE}) when switched on, and therefore a voltage drop. [*The resistance is a result of the PN junction doping process during manufacturing.*]

In addition to the transistor voltage drop, the LED will also drop somewhere between 1.2 and 3 volts when it's switched on (check the datasheet under V_F). Therefore, in order to calculate the correct value for resistor R_C, the voltage drop across the collector/emitter ($V_{CE(sat)}$) and the voltage drop across the LED ($V_{(LED)}$) must be included in the formula. So, here's the same Ohm's Law formula modified to account for all the voltage drops:

$$R_C = \frac{V_{CC} - V_{(LED)} - V_{CE(sat)}}{I_{C(MAX)}}$$

Now, let's calculate the correct resistance value for resistor Rc. Let Vcc = 5V, $V_{(LED)}$ = 1.9V, $V_{CE(sat)}$ = .1V, and $I_{C(MAX)}$ = 15 mA. The answer is:

$$R_C = \frac{V_{CC} - V_{(LED)} - V_{CE(sat)}}{I_{C(MAX)}}$$

$$R_C = \frac{5 - 1.9 - .1}{.015}$$

$$R_C = 200 \text{ ohms}$$

The calculation shows that we need a 200 ohm resistor for Rc in order to limit the current through the LED to a safe 15 mA. Notice, had we used the basic Ohm's Law formula ($R_C = V_{CC} / I_{C(MAX)}$), RC would be 333.33 ohms. The real problem with using a 333.33 ohm resistor for RC begins when you actually breadboard the circuit, only to find out the current you expected through the LED is not the required 15 mA, but 9.2 mA (a 39% loss). Therefore, if you fail to add both the LED and transistor voltage drops in the calculation, your LED won't be as bright as expected.

Try and look at the LED and transistor as small resistors. In a series circuit, you would add all the resistor values together to get the total resistance, right? Well, all we're doing here is accounting for all the voltage drops in a series circuit.

Figure 3 clearly shows what happens to the collector current ($I_{C(MAX)}$) when you don't include all the voltage drops in the formula.

BASE TO CONTROL

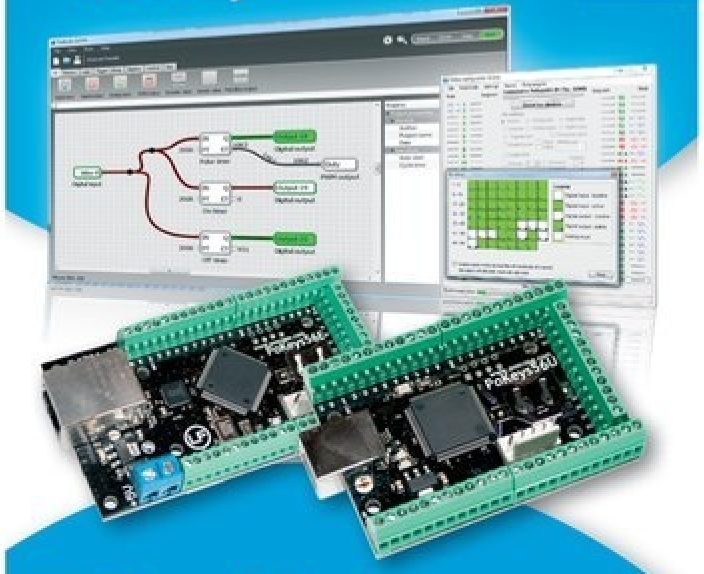
The question now is how do you control the transistor so it turns on and off? Well, we have to do two things: 1. Find the correct transistor base current (I_B) that will saturate the transistor. 2. Calculate the resistance value for the base resistor R_B (see **Figure 1**). The formula for finding the base current is:

$$I_{B(EOS)} = \frac{I_{C(MAX)}}{Beta\ (min)}$$

Notice here, in order to find the base current (I_B), we divide the maximum collector current ($I_{C(MAX)}$) we want to go through the LED (15 mA) by the minimum *B*eta listed on the datasheet (h_{FE}). What is *B*eta? *B*eta — also known as DC current gain — is a ratio relating to how much current gain you can expect through a transistor's collector terminal given a certain amount of current going into the base terminal. In other words, the base current controls the collector current. It's kind of like a small water valve controlling the flow of water running through a large pipe.

Having said all that — and this is very important — ***Beta (gain) is only used in amplifier design***. When you're using a transistor as a switch (digital mode), *B*eta has little effect or meaning because the transistor is not operating in the active region that amplifiers work in. Once a transistor switch is in saturation mode, there's no collector current gain beyond saturation.

In other words, once a transistor switch reaches the saturation point, the gain formula IC = *B*eta x I_B no longer applies because the voltage drop across the collector/emitter terminals ($V_{CE(sat)}$) has reached its lowest saturation voltage of .1V. When $V_{CE(sat)}$ reaches this voltage



Facts about the Transistor Switch

1. Any level of collector current (Ic) in between the two states of saturation and cut-off is not important to the design or functioning of a transistor switch — it's only important to amplifier designers.

2. When using a transistor as a switch (digital mode), DC *B*eta (h_{FE}) has no meaning because the transistor is not operating in the active region that amplifiers work in. A transistor switch is either in saturation mode (fully on) or cut-off mode (fully off). In other words, the gain formula Ic = *B*eta x Ib is invalid beyond the saturation point.

3. The saturation current ($I_{c(sat)}$) flowing through a transistor switch is not determined by the transistor's internal electrical parameters, but rather by the external components (resistor/LED) employed by the circuit designer.

4. To force a transistor switch into deep saturation, the circuit designer adds an overdrive factor to the base current.

level, the collector current can't increase beyond this point — *even if the base current continues to increase.*

Remember, a transistor operating in digital mode (on/off) is either in saturation mode (fully switched on) or in cut-off mode (fully switched off). Therefore, any level of collector current (Ic) in between the two states of saturation and cut-off is not important to the functioning of a transistor switch — it's only important to amplifier designers.

Okay, so what value do we use for *B*eta in the formula to find the base current (I_B)? Well, the standard rule of thumb states that you should use the minimum *B*eta (h_{FE}) listed on the datasheet. Unfortunately, the minimum *B*eta listed on the datasheet will only place the transistor at the Edge of Saturation (EOS). Since transistors are sensitive to temperature changes, a change in temperature could force the transistor to move from the EOS into the "active" area (amplifier region).

Therefore, in order to eliminate this possibility, we use what is known as an "Overdrive Factor" (ODF). This is an arbitrary number between 2 and 10 that is used to insure that the transistor is driven hard into saturation (fully turned on) — and where temperature changes fail to drop the transistor out of saturation. Therefore, IB equals:

$$
\begin{aligned}
I_B &= I_{B(EOS)} \quad \times \; ODF \\
& \qquad \downarrow \\
I_B &= \frac{I_{C(MAX)}}{Beta\ (min)} \quad \times \; ODF \\
I_B &= \frac{.015}{100} \quad \times \; ODF \\
I_B &= .15\ mA \quad \times \; 10 \\
I_B &= 1.5\ mA
\end{aligned}
$$

Notice, in the formula above, by using an ODF of 10 we increase the base current from 150 µA to 1.5 mA, thereby assuring that the transistor is forced into deep saturation. For example, if a datasheet listed a *B*eta(min) of 75, and you needed a collector current ($I_{C(MAX)}$) of 25 mA, I_B would be .333 mA (.000333A). Unfortunately, 333 µA would only put the transistor at the EOS. By using an ODF of 10, we increase the base current (I_B) to 3.3 mA — well beyond the EOS and into deep saturation.

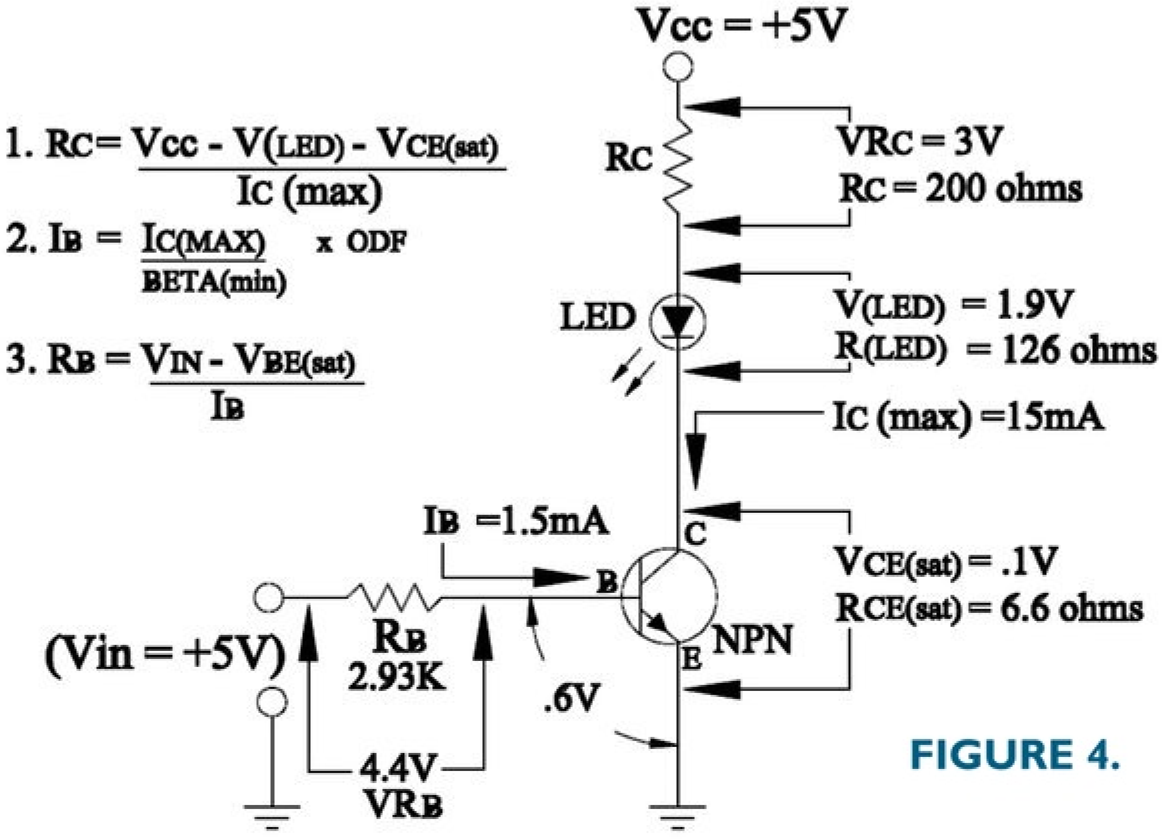
Now that we have established a base current (I_B) of 1.5 mA is required to saturate our transistor, let's calculate the resistance value needed for the base resistor R_B. Once again, we use Ohm's Law to calculate for R_B:

$$
\begin{aligned}
R_B &= \frac{V_{IN} - V_{BE(sat)}}{I_B} \\
R_B &= \frac{5\ -\ .6}{.0015} \\
R_B &= 2933.33\ ohms
\end{aligned}
$$

Note in the formula above, that $V_{BE(sat)}$ is the required base voltage that must be present in order to forward-bias the transistor's base/emitter junction (i.e., to turn the transistor on). Generally speaking, this value is between .6 to .7 volts for a general-purpose transistor. Always check $V_{BE(sat)}$ listed on the datasheet to verify.

Finally, we calculate the voltage drop across R_B (V_{RB}) by multiplying the base current (1.5 mA) by the resistance of R_B (2.933K). Therefore, V_{RB} = 4.4V.

Figure 4 shows the finished LED circuit with all the components and electrical parameters clearly marked (Ohm's Law was also used to calculate the resistance for $R_{(LED)}$ and $R_{(CE)}$).



1. $R_C = \frac{V_{CC} - V_{(LED)} - V_{CE(sat)}}{I_{C\,(max)}}$
2. $I_B = \frac{I_{C(MAX)}}{BETA(min)} \times ODF$
3. $R_B = \frac{V_{IN} - V_{BE(sat)}}{I_B}$
Vcc = +5V
Rc
VRc = 3V
Rc = 200 ohms
LED
V(LED) = 1.9V
R(LED) = 126 ohms
Ic (max) =15mA
IB =1.5mA
C
B
E
NPN
VCE(sat) = .1V
RCE(sat) = 6.6 ohms
(Vin = +5V)
RB
2.93K
.6V
4.4V
VRB

FIGURE 4.

We now have the correct resistor values in order to operate the LED and transistor circuit in a safe manner: Rc = 200 ohms; and R_B = 2933.33 ohms.

I'm sure you've notice that our 2.933K resistor is not a standard size you can actually purchase anywhere. The rule of thumb in this case states that you can use the next standard resistor value *below* 2.933K (2.7K to 2.87K). Why?

The lower resistance only helps to decrease the chance of the transistor from falling out of saturation mode during temperature and power supply variations by increasing the base current (i.e., transistor goes even deeper into saturation).

RECAP

Let's review all the steps required to use a transistor as a switch:

1. Download the datasheets for the LED and transistor.

2. Determine the max current ($I_{C(MAX)}$) you want to go through the LED and transistor, and verify that it doesn't exceed the maximum current rating of the LED (I_F) or the transistor (Ic); refer to the datasheet.

3. Calculate the value for resistor Rc. Make sure to include the voltage drops for the LED ($V_{(LED)}$) and the transistor ($V_{CE(sat)}$) in the Ohm's Law formula.

4. Calculate the transistor's base current I_B using an ODF of 10.

5. Calculate the resistance value for base resistor R_B.

That's it. Kind of easy — well, maybe not.

FINAL NOTE

On a personal note, when I breadboard a circuit I only use through-hole, red, 5 mm diameter, ultra-bright, water-clear dome 640 nm LEDs. I've tried other LEDs, but the water-clear dome LEDs are the best. They're so bright, they hurt your eyes — no kidding!

Order a bag of LEDs from Digi-Key, Jameco, or Mouser for your next project — it makes life easier. **NV**

Fix Up that Old Radio!

By J.W. Koebel

It's fun – and easy – to bring your vintage radio back to life!

Maybe you were digging around in the attic helping your parents downsize, and found their old wooden tube radio from many years ago stashed away. Or, perhaps you've inherited your grandparent's precious tube console — now a family heirloom — that stood as the centerpiece of their living room for many years. Even though they're ancient by today's standards, these old tube radios can almost always be brought back to life, and there's still plenty of broadcasts out there for them to receive! Not to mention, they just look so classy and have a presence few modern electronics can match.

Fixing up a vintage radio is a little different from fixing up many other types of gear, since the entire field of electronics had only just been invented when they were first made. We were still learning the engineering and physics underlying all this new technology.

Competing standards emerged and fell by the wayside, and huge advances were made in the span of only a few years — not to mention some vicious patent wars — which all mean that no two radios are alike.

Even still, since radio was the only means of home entertainment for decades, there's plenty of service information out there to help you through the process, and there's nothing quite like the feeling of hearing a piece of history crackle back to life after a successful repair job.

Over a series of articles, I'll be performing a full electrical restoration on a 1937 DeWald Model 618 vintage radio from start to finish: evaluating its condition, performing the restoration work, and aligning and verifying its performance after service is complete. In this first article, I'll cover some tips for what to look for on your old radio to judge its condition, and do a full inspection of the internals to see what work will need to be done.

First, a Short History Lesson

Vacuum tubes operate on the principle of thermionic emission, where certain metals heated to white-hot in a vacuum will give off electrons. The flow of these electrons

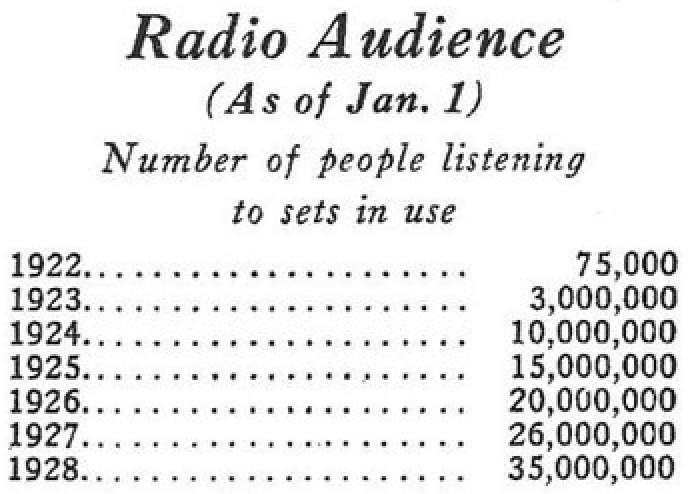
The Radio Set You Want at the Price You Want to Pay

A 5-Tube Set Superior to Most 6 and 7-Tube Sets in Tone Quality, Volume and Distance Getting

An early radio with multiple controls; the Barawick Five (*Popular Mechanics*, November 1926).

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com /index.php?/magazine/article/may2015_Koebel.**

Radio's listening audience grew rapidly in the early days (*Radio Retailing*, March 1928 via "Radio Manufacturers of the 1920s" by Alan Douglas).



Radio Audience

(As of Jan. 1)

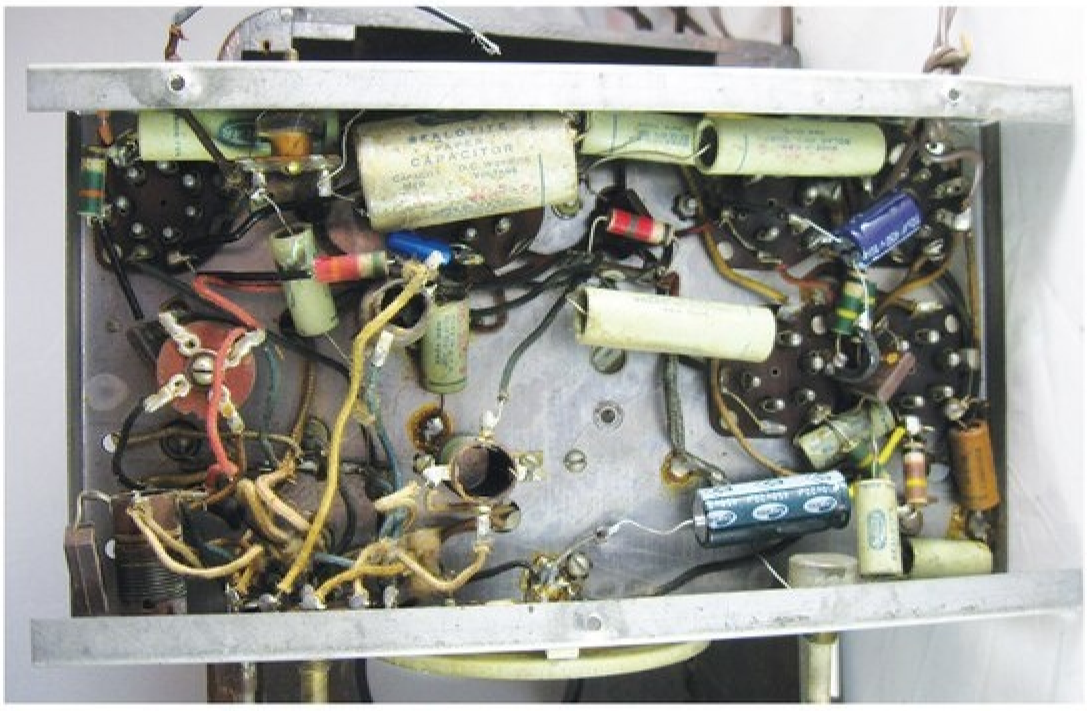
Number of people listening to sets in use

1922	75,000
1923	3,000,000
1924	10,000,000
1925	15,000,000
1926	20,000,000
1927	26,000,000
1928	35,000,000

can then be controlled by charges placed on the tube's internal elements. The first practical vacuum tubes were invented in 1907 by Lee DeForest, who developed the "Audion" — the world's first triode capable of amplification. This ushered in the radio age, but it wasn't until about 20 years later that radio finally left the lab and made it to consumer's homes.

Even then, most radios still operated on stacks of expensive batteries which had to be replaced often to deliver the high voltages necessary to operate. It wasn't until widespread electrification of the United States and the development of tubes which could be heated with AC power instead of DC at the very tail end of the 1920s that radio exploded and achieved mass-market status.

Stacks of batteries disappeared and radios went from being pieces of lab equipment with many controls to adjust independently to the format we've retained today. Point the needle to the station on the dial and off you go.



Underside of the radio showing mostly original capacitors, but with a few more recent replacements.

Caution!

Electronics of the era were really brute-force devices, and tube radios were no exception. **It is incredibly important you follow proper safety precautions while working on any tube radio as they present some different hazards than most other hobbyist projects which run on a dozen or so volts.** Even an entry-level radio from the '20s or '30s might have as much as 500V DC (with 50-100 mA of current available!) on the tube's plates in the circuit.

Safety standards were nearly non-existent, and it was common to find certain transformerless "AC/DC" radios with one side of the incoming power connected to the chassis. In a modern household system, this could place 120V AC on any exposed metal bit.

In addition to all these electrical hazards, tube radios get quite hot while operating, and you can easily get a serious burn by grabbing a tube before it's cooled. Always work with the radio unplugged, and when testing a powered-on radio, use one hand. If you're working on a hot chassis radio, make sure to use an isolation transformer or you could be badly injured or killed. Go slowly, double-check your work, and be careful!

If you find a radio in the attic, you might be tempted to plug it in and see if it works before getting started. **Do not ever do that!** Signal routing in these vintage radios was primarily done with capacitor coupling, and the capacitors back then were literally made out of cardboard, foil, paper, and wax which badly degrade with age.

Odds are that if you try and plug in a radio you've just found, not only will it not work, but shorted capacitors will damage expensive and difficult to replace components like transformers, or even start a fire. It only takes a few seconds. Never power-on a vintage radio without inspecting it, and at a minimum replacing the capacitors!

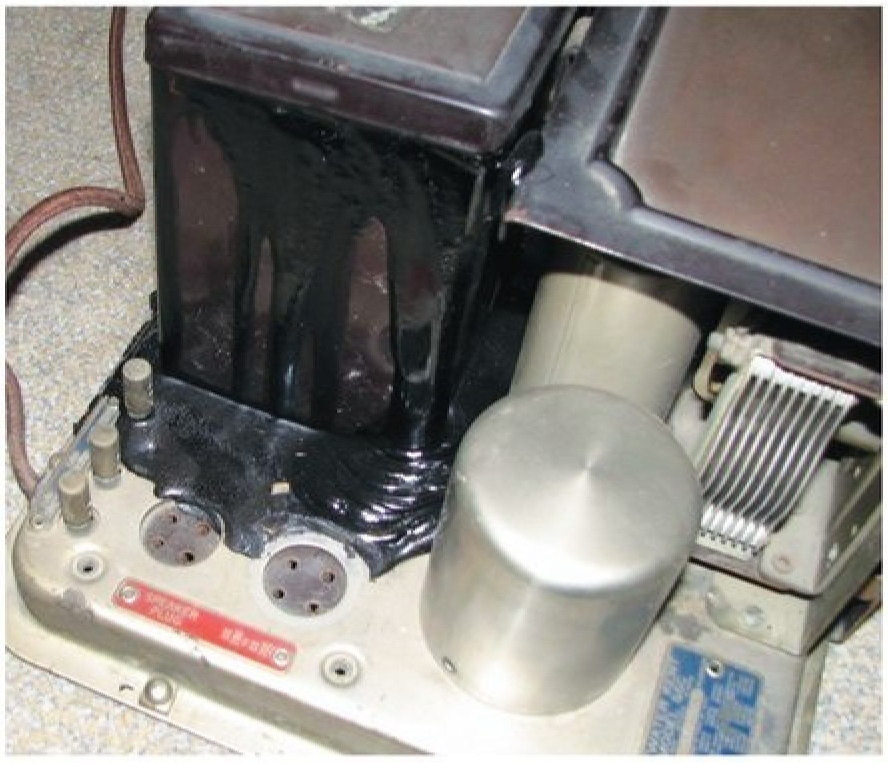
Finally, if you've found a project

Unlike transformer-operated radios which are inherently isolated, many more economical radios saved the cost of the transformer and connected the radio's circuitry directly across the AC line. This can result in a significant safety hazard as now the radio's metal components may become "hot" relative to ground. You could receive a fatal shock by accidentally contacting a cold water pipe or damp concrete floor, or destroy any grounded test equipment you connect to the device under test.

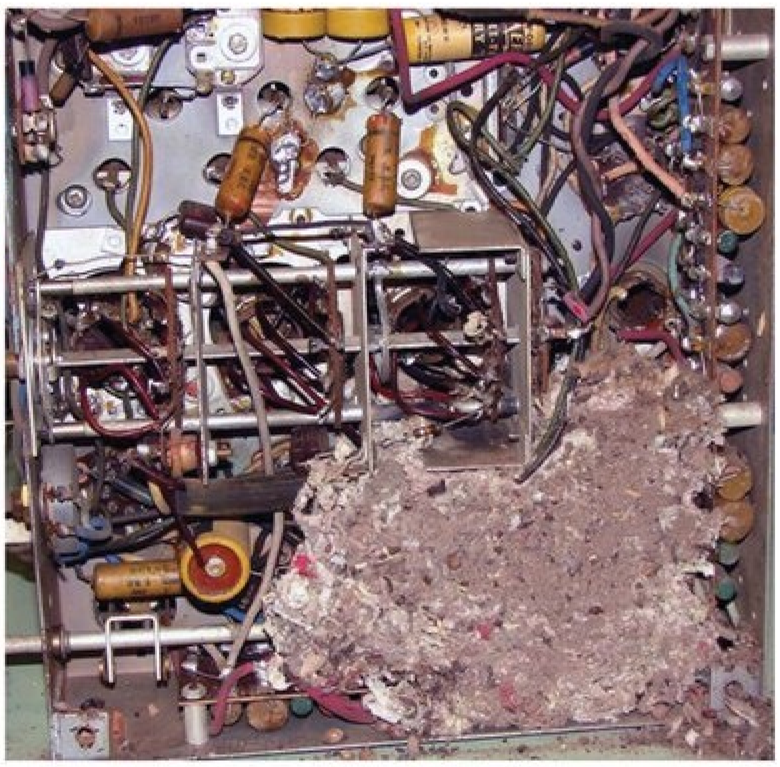
An isolation transformer will interrupt the current path to earth ground and provide safety in these circumstances — both for yourself and your equipment. Most isolation transformers are 1:1, although tapped models with provisions for slightly adjusting the line voltage (i.e., 120:115-120-125) and continuously variable models do exist. Don't use a variac, though — they're not isolated and won't provide any safety benefits — unless you have a separate isolation transformer to go with it.

This 1931 Westinghouse grandfather clock radio which was previously on fire wouldn't be a good starter project.

The rear of the DeWald 618, with the rare original back intact.



This radio had shorted filter capacitors in the power supply which caused overcurrent, overheating, and melting in the power transformer.



A vintage radio chassis infested with mice. Not a good candidate for repair!

radio to start on with more than six or seven tubes, you might want to find a smaller model to practice on before diving in — especially if the radio has any sentimental value to you. The number of tubes gives a good estimate of the circuit complexity, and even if you're skilled at electronics they're different enough from most projects that it's good to get a feel for how tube circuits work before starting on that special radio.

So, What Do I Need?

For most radios, you don't need much at all. It's entirely possible to do a full radio restoration with nothing but a multimeter, a soldering iron, and your standard assortment of screwdrivers, pliers, and clippers. If you're working on a "hot chassis" radio which lacks a main power transformer, you'll want to use an isolation transformer. If you're a perfectionist, you may want a modulated signal generator to help out with the alignment. In general, these are simple pieces of electronics and you don't need much to be successful. That's the best part — no special tools required!

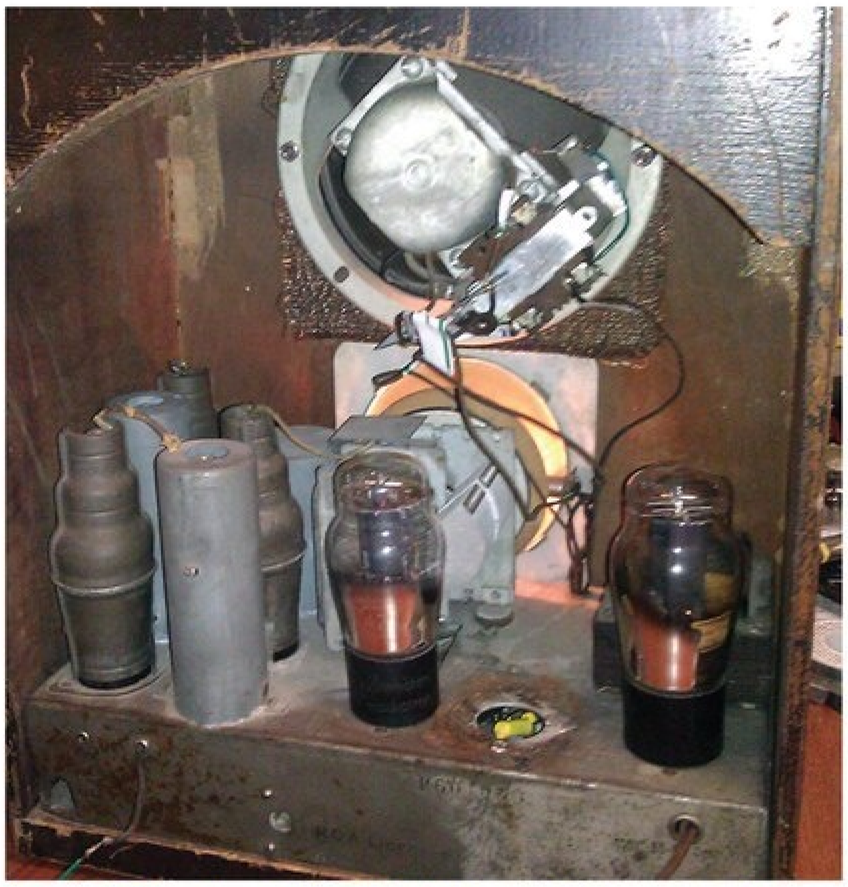
Most of the components you'll use can be obtained from everyday electronics parts houses like NTE PartsDirect or Mouser. If you're just starting out — or just don't care to wade through some of those more complicated sites to find what you're looking for — specialty retailers like New Sensor (**www.newsensor.com**), Sal's Capacitor Corner (**www.tuberadios.com**) or Just Radios (**www.justradios.com**) have a more focused selection of parts specifically for these kinds of projects.

More often than not, the tubes that are already installed in your radio are good and don't need to be replaced. If they do, though, you can find most tubes at Antique Electronics Supply (**www.tubesandmore.com**). There's a thriving hobbyist community with thousands of active members from around the world (myself included!) over at the Antique Radio Forum (**www.antiqueradios.com**) to help you out if you need to find something specific and/or rare, or if you'd just like to chat about your new project.

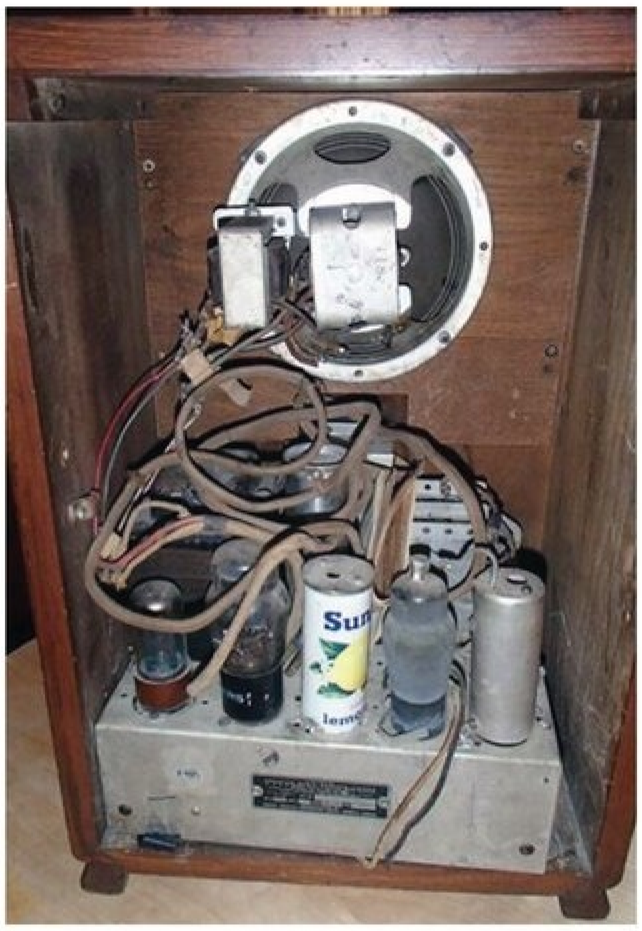
Basic Visual Inspection

If you've already found a radio, now it's time to check it out a bit. I'll be using my DeWald 618 as an example. First, take a good look at the back of the cabinet. Do you see any signs of smoke damage? If so, that would be a big red flag. Fortunately, it's pretty rare to find one that's been on fire previously, so most of the time you'll be looking for more subtle problems.

These radios have been around a long time and were often in the shop every few years when they were new, so it's worth checking for a few things. If your radio has a large power



This Simplex Model P tombstone radio has an extra bolted-on output transformer, and is missing a filter capacitor can.



Someone replaced an IF transformer on this 1935 Rogers radio with a juice can.

transformer, check to see that it hasn't overheated and melted the potting tar out from the insides. This generally means a burned out transformer and a dead radio, and is one of the most common outcomes if you plug in a radio you've found without refurbishing it first.

Keep an eye out for evidence of rodent or insect infestations. Bite marks and chewed wiring are a major warning sign. Generally, a radio that's been infested isn't worth even attempting to fix due both to the damage that may exist and the fact it's a serious health hazard.

Finally, keep an eye out for modifications made sometime in the past. Empty holes on the chassis, extra bolted-on parts, or things that just don't belong are evidence of someone having been in there before. That's not necessarily bad, but it means you'll need to carefully check the radio out.

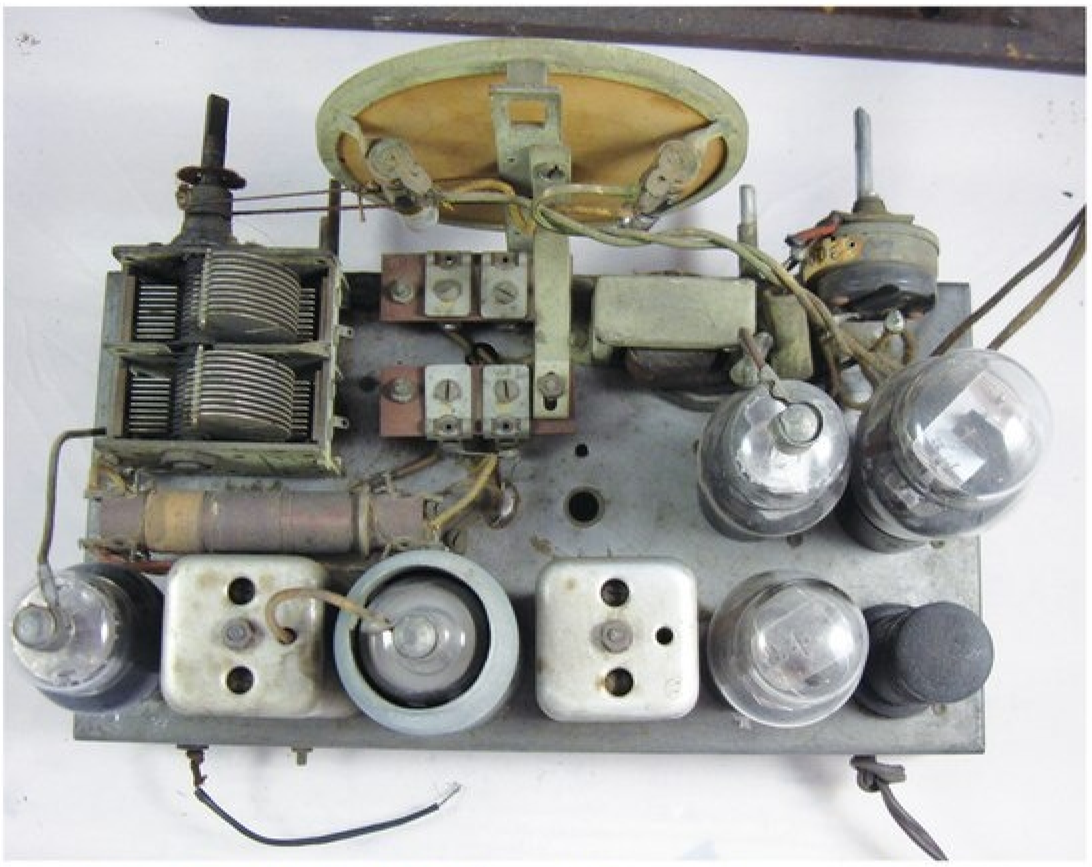
There are two kinds of radios: superheterodyne or "superhet;" and tuned radio frequency or "TRF." These relate to differences in the way they process the radio signals, but all share mostly the same set of parts.

There's a power supply which typically had one or two inductors for filtering. From the antenna, there will be a set of antenna coils which are radio frequency (RF) transformers used in pre-selecting the tuned station. There's further sets of RF coils depending on the radio's topology: more RF coils for a TRF, or a set of oscillator coils and intermediate frequency (IF) transformers.

Finally, there's the audio output section which will have an output transformer which drives the speaker, and occasionally an interstage transformer as the input to the audio amplifier stage. All of these coils and transformers need to be working for the radio to receive correctly.

Principles of Operation

The DeWald 618 is a superhet. In this five-tube AC/DC radio, the antenna picks up the transmissions and couples them through a set of antenna coils which are switched depending on whether you're listening to mediumwave broadcast signals or one of the shortwave bands, and help to pre-select the signal going into the next stage of the radio to reduce interference and undesired reception.



The DeWald 618 chassis from the top down.

This signal is fed into the 6A7 tube — the pentagrid converter which serves the double function of both local oscillator (driven by a set of oscillator coils) and mixer.

After mixing the RF signals together, the converter outputs the received signal on the radio's IF. This IF signal is first coupled through an IF transformer, tuned to the intermediate frequency, and into the 6D6 IF amplifier tube.

After being amplified, the signal exits the plate of the IF amplifier and travels through another IF transformer to the #75 tube which is the combined detector, audio amplifier, and automatic volume control level generator. The diodes in the #75 tube detect (rectify) the modulated RF signal, into wave riding on top of a DC offset.

In most radios, this DC level is proportional to the received signal strength and is routed to the radio's front-end tubes as a part of the automatic volume control (AVC) circuit. This helps keep the radio's volume constant as you tune around the dial and as the strength of the radio signal naturally fluctuates due to atmospheric conditions. The audio riding on top of the DC is passed through a capacitor and the volume control, and into the control grid (input) of the #75 where it exits the plate and drives the #43 output tube. The signal travels from the #43 through an output transformer and reproduces the program's audio to your ears via the speaker.

In-Depth Intake Checks

Before getting started with the labor of replacing components, it's useful to check the status of all the coils and transformers. These parts are all called out on the **schematic**.

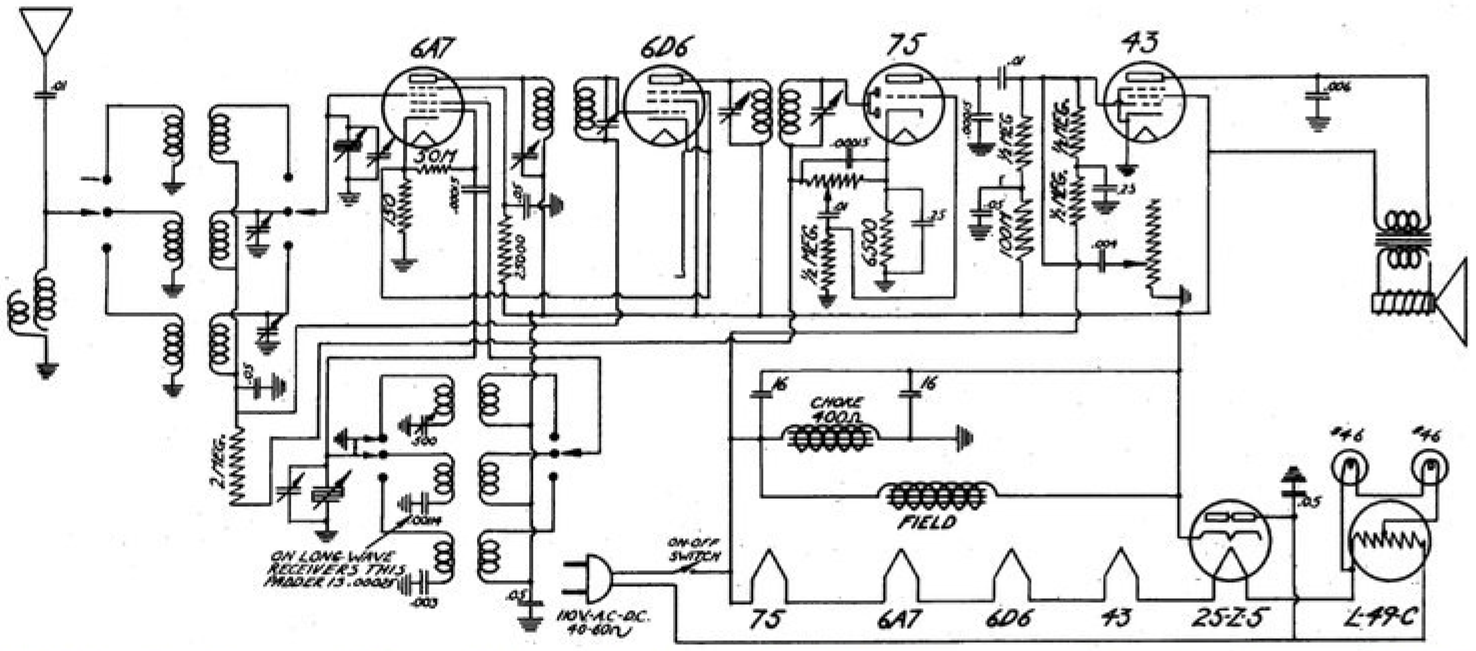
After pulling the chassis and speaker from the cabinet, the first step is to remove the tubes. Make a note of which tube goes in which socket since they're not interchangeable. Some radios (like the one I'm using here) have the tube number stamped into the sockets — if that's the case, you're lucky!

If any of the tubes have a top cap with a wire connection, be careful when removing it — sometimes the top caps of the tubes have come loose with age and can be pulled off which often ruins the tube.

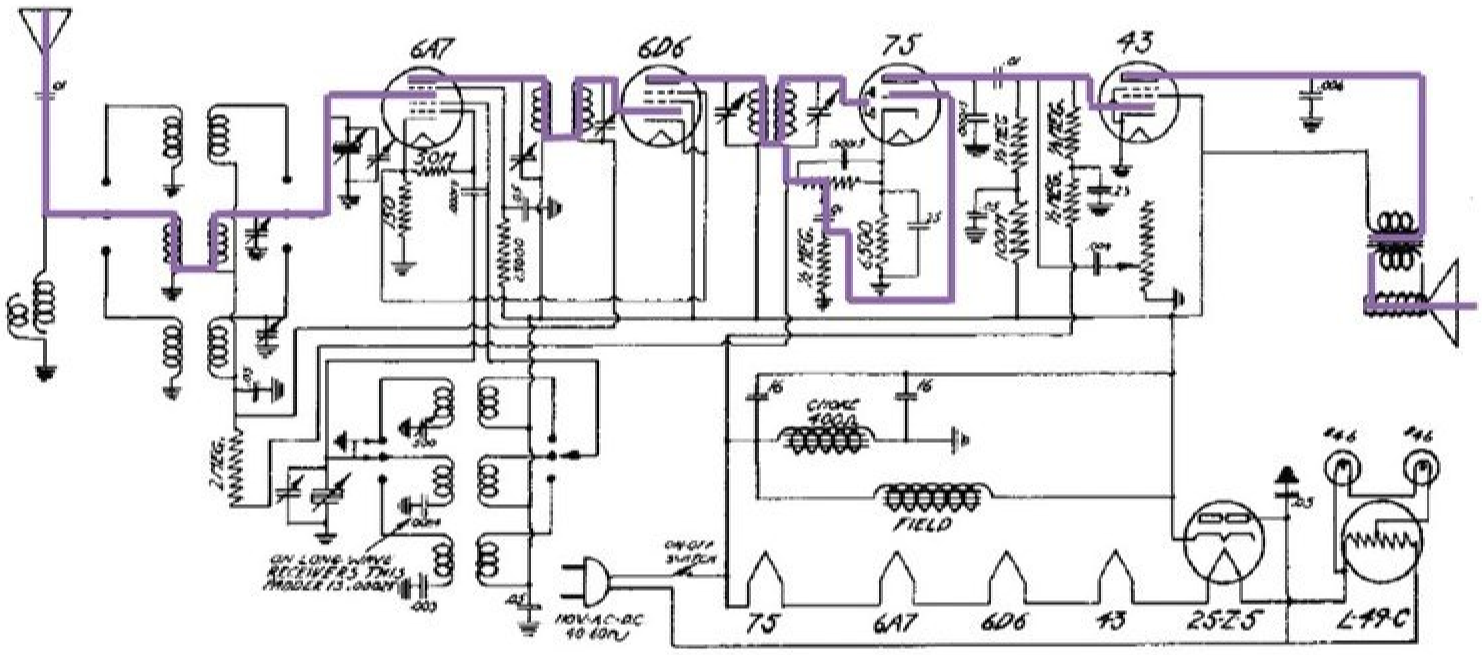
With the tubes removed, now you can access the various connections a bit more easily. I like to start with the radio's front end with the antenna coils, and move through the radio to the output transformer. To check their condition, simply use a multimeter set to measure resistance (ohms) and check for continuity.

An infinite resistance reading means there's a broken wire, but good readings might range from a few ohms on the low end to a few dozen on the high end.

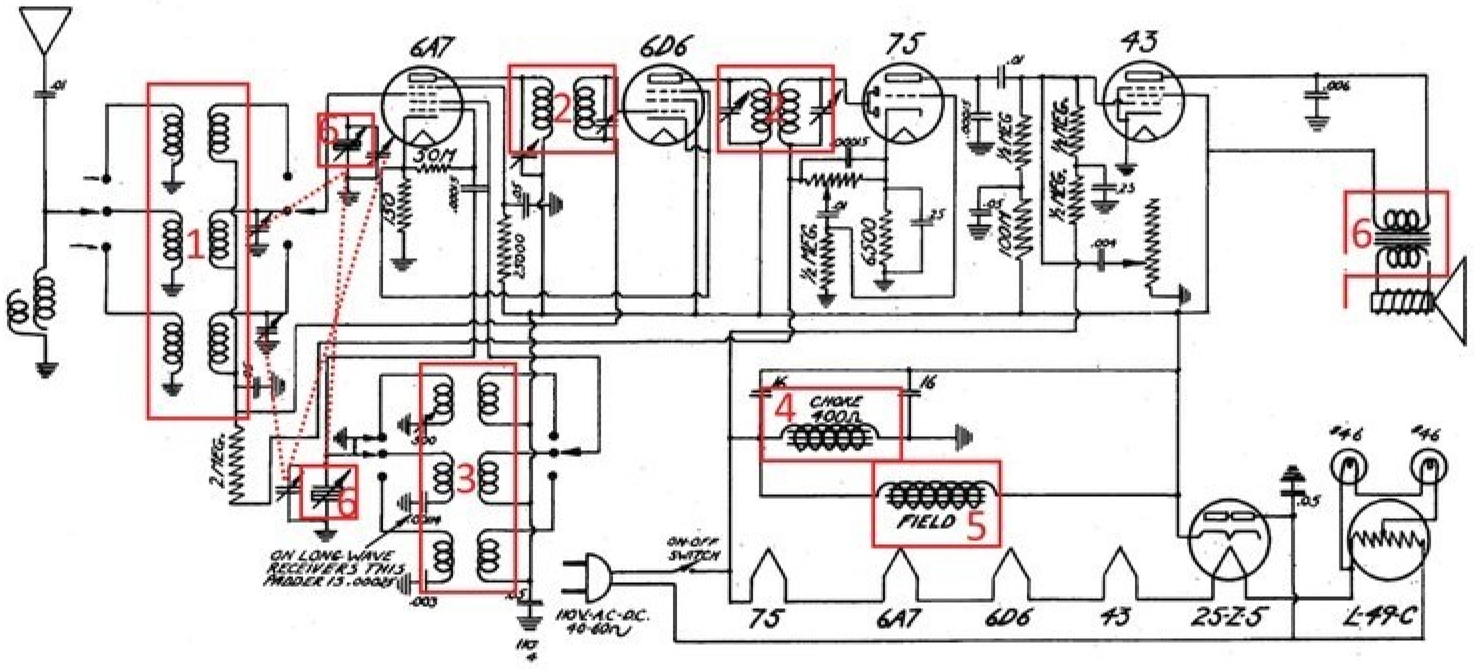
One good trick for measuring the primary of the antenna coils is to take advantage of the radio's built-in wiring. Since the antenna connects to all of these coils and all of these coils connect to ground after the band switch, connect the multimeter between the antenna terminal on the back and ground, and flip through



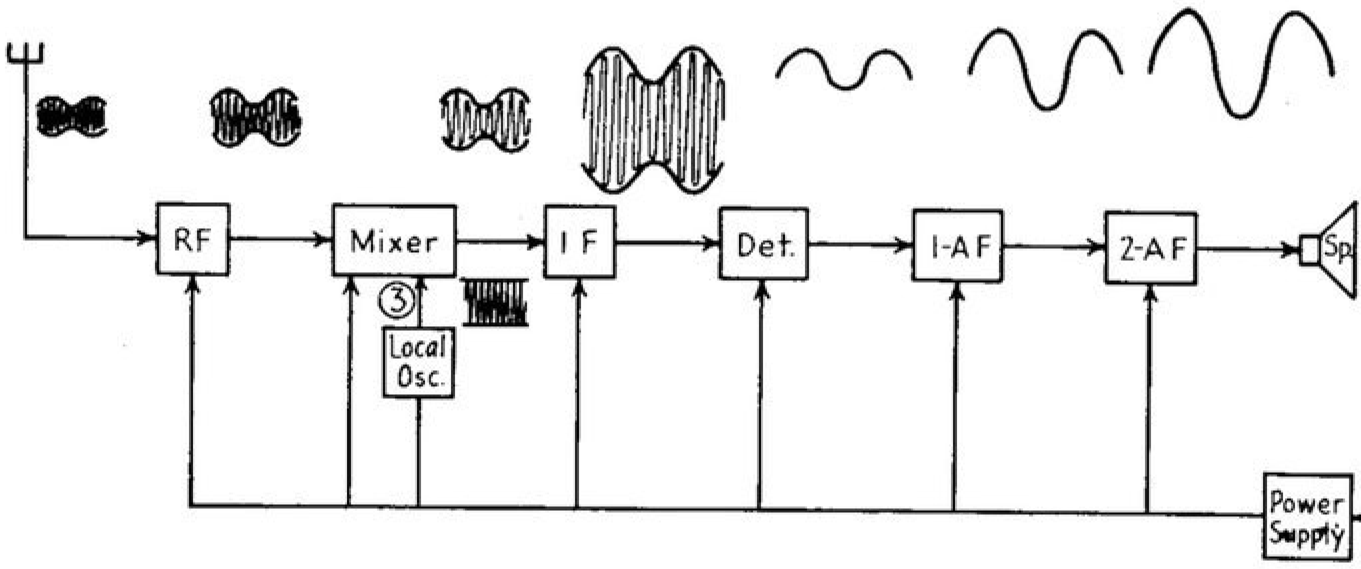
Original schematic for this radio from *Rider's Perpetual Troubleshooter Vol. 7.*



The schematic showing the signal path from the antenna to the speaker.



Components as marked: (1) Antenna coils; (2) IF transformers; (3) oscillator coils; (4) filter choke; (5) field coil; and (6) output transformer.



Block diagram showing the signal path through the radio, with wave shapes shown (*Elements of Radio Servicing*, Marcus and Levy, 1947).

the ranges. In this case, all good! I measured the secondaries of the antenna coils as well, and they also checked out. On to the oscillator coils!

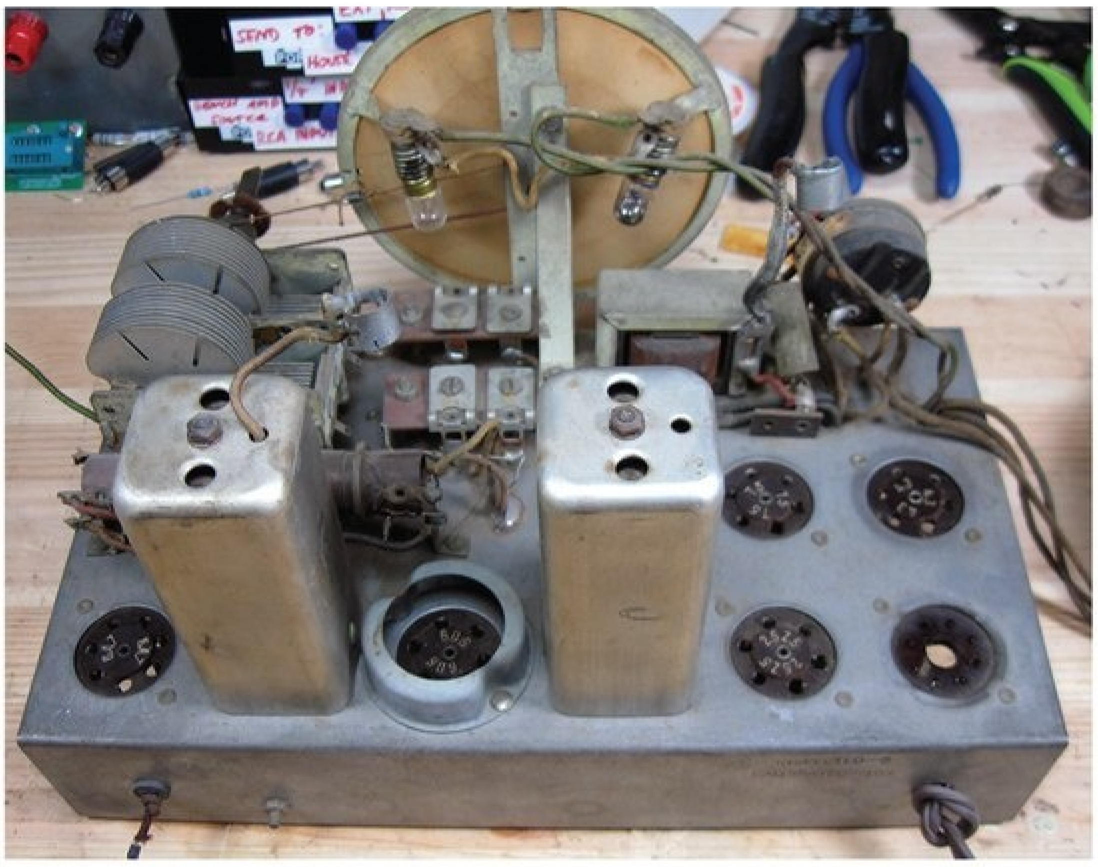
This radio has three oscillator coils total: one on its own coil form beneath the chassis, and two wound on the same coil form and mounted up next to the variable capacitor on the top. These are each four-terminal devices, so they're easy to check out. There will be two sets of lugs with continuity to each other, but not to the lugs from the opposite winding. In this case, all oscillator windings tested good too.

Next up are the IF transformers. These cans are mounted to the chassis, and each has two windings: a primary and a secondary. It's fairly straightforward to trace out the wiring and find two pairs which have continuity.

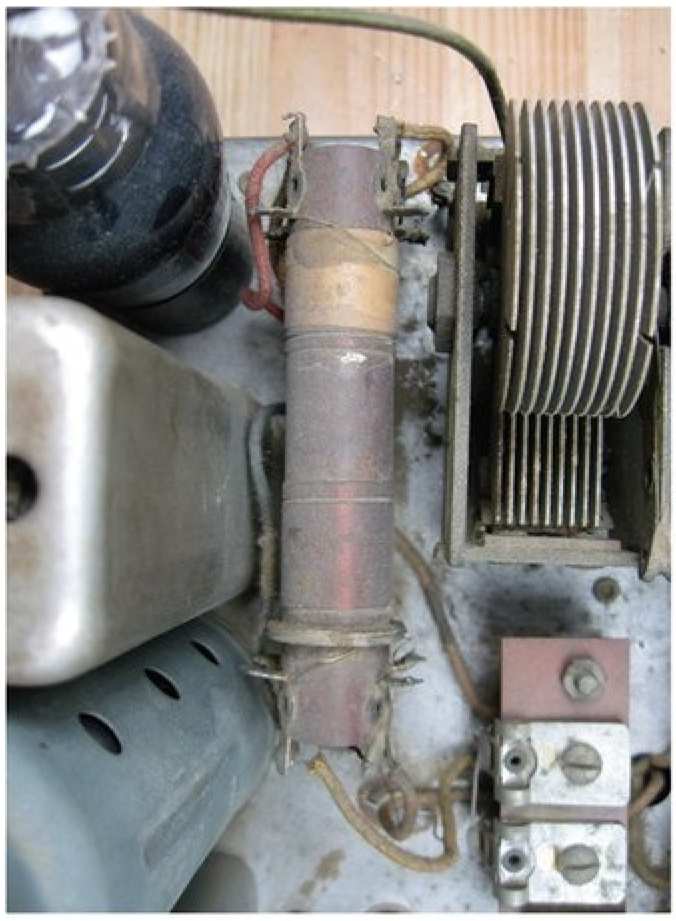
On the first IF transformer, the secondary has one connection out the top to the grid cap of the 6D6 IF amplifier tube. On the second IF transformer, they're both on the bottom. After checking continuity on both, the IF transformers are also good.

Some more advanced radios might have tapped IF transformers or other components hidden inside the cans, but that's not something you're likely to find on an entry-level radio like this one.

Moving on, since this is a transformerless AC/DC radio, the only items left to check are the filter choke, the output transformer, and the field coil. The filter choke sits on top of the chassis, with the leads heading towards the power supply section. There's only one winding, and the approximate DC resistance is given on the **schematic**. These tended to have a ±20% tolerance, so the reading of 345 ohms isn't indicative of any problems.

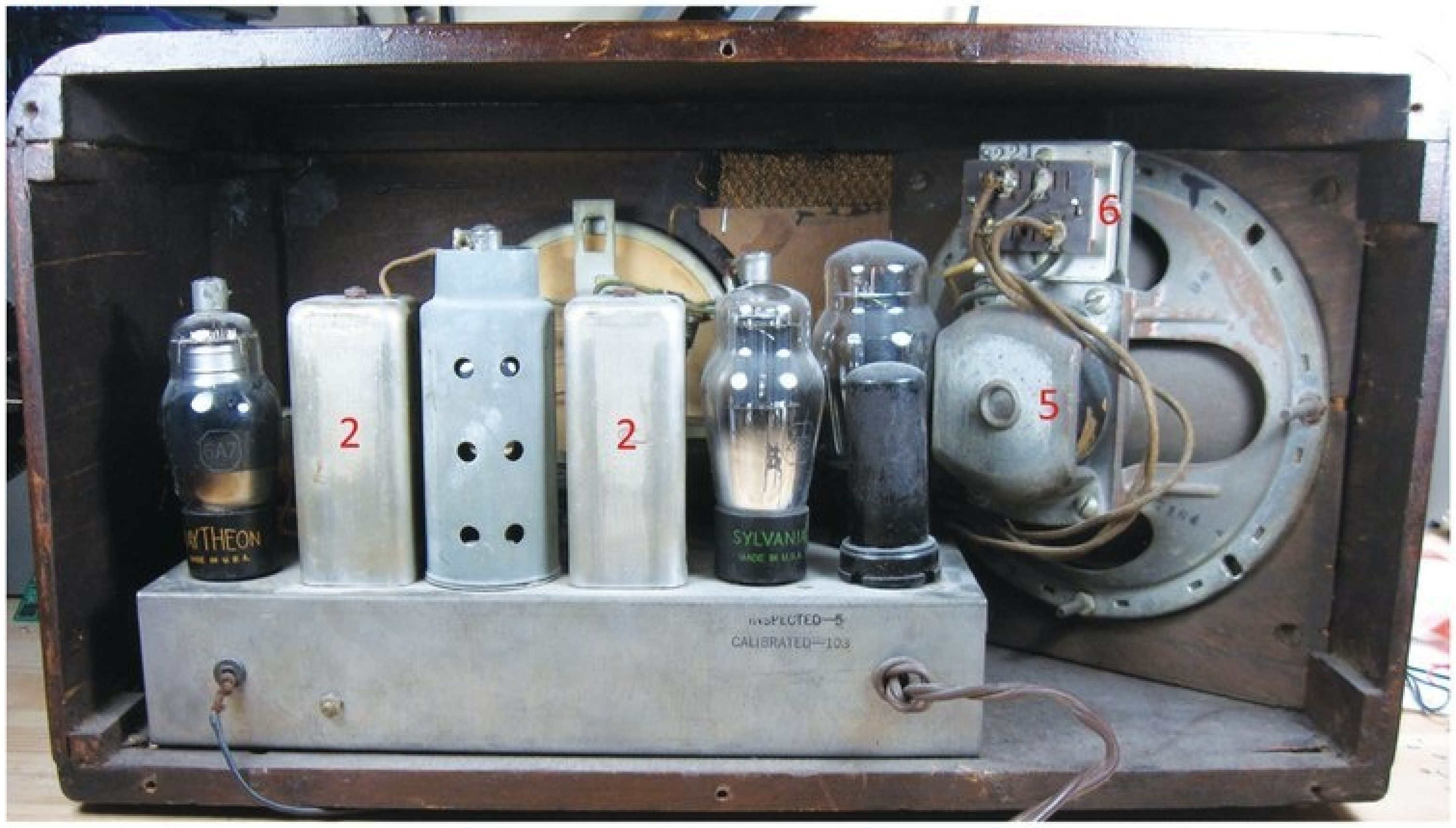


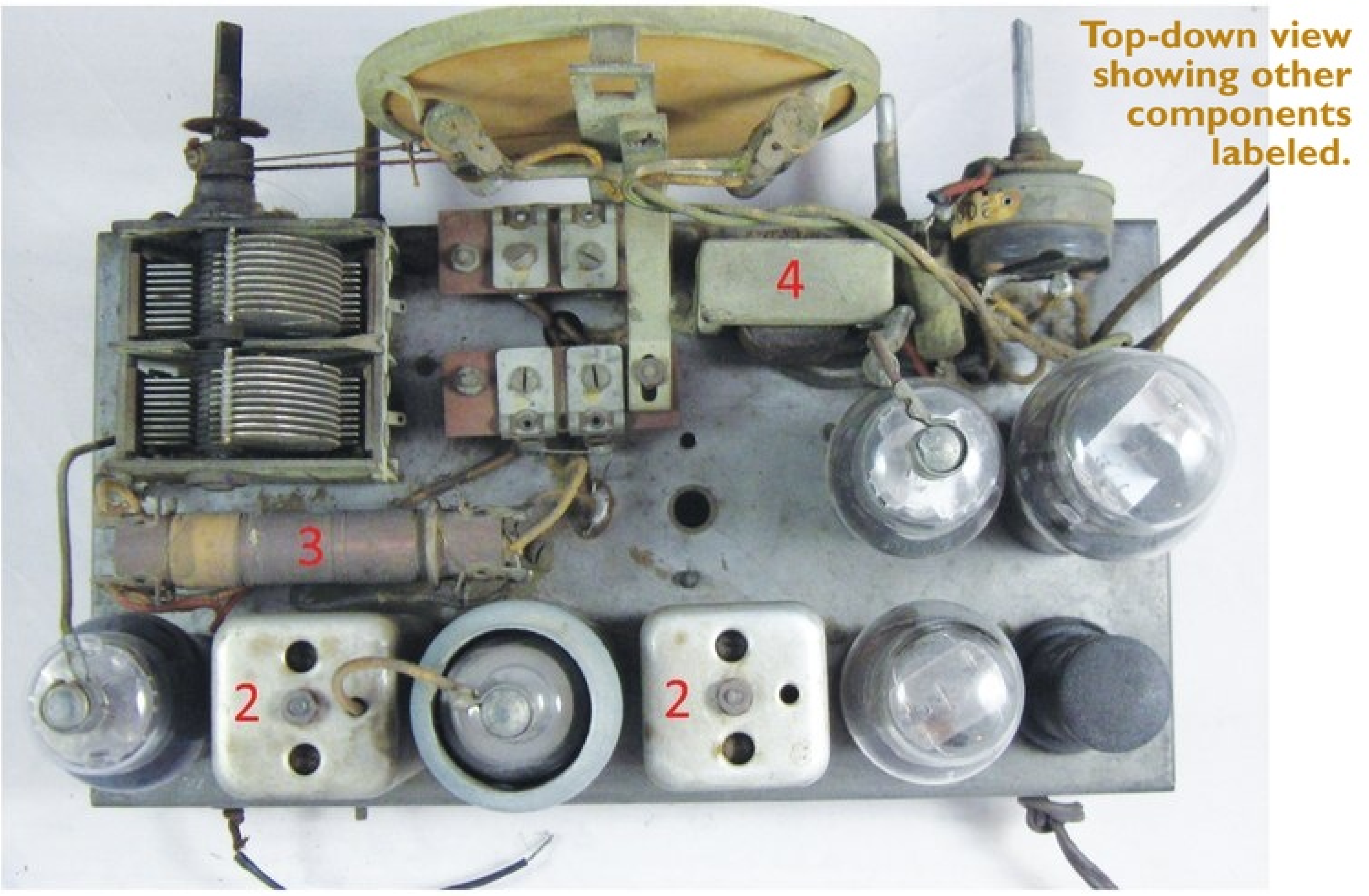
Bare radio chassis with tubes removed.



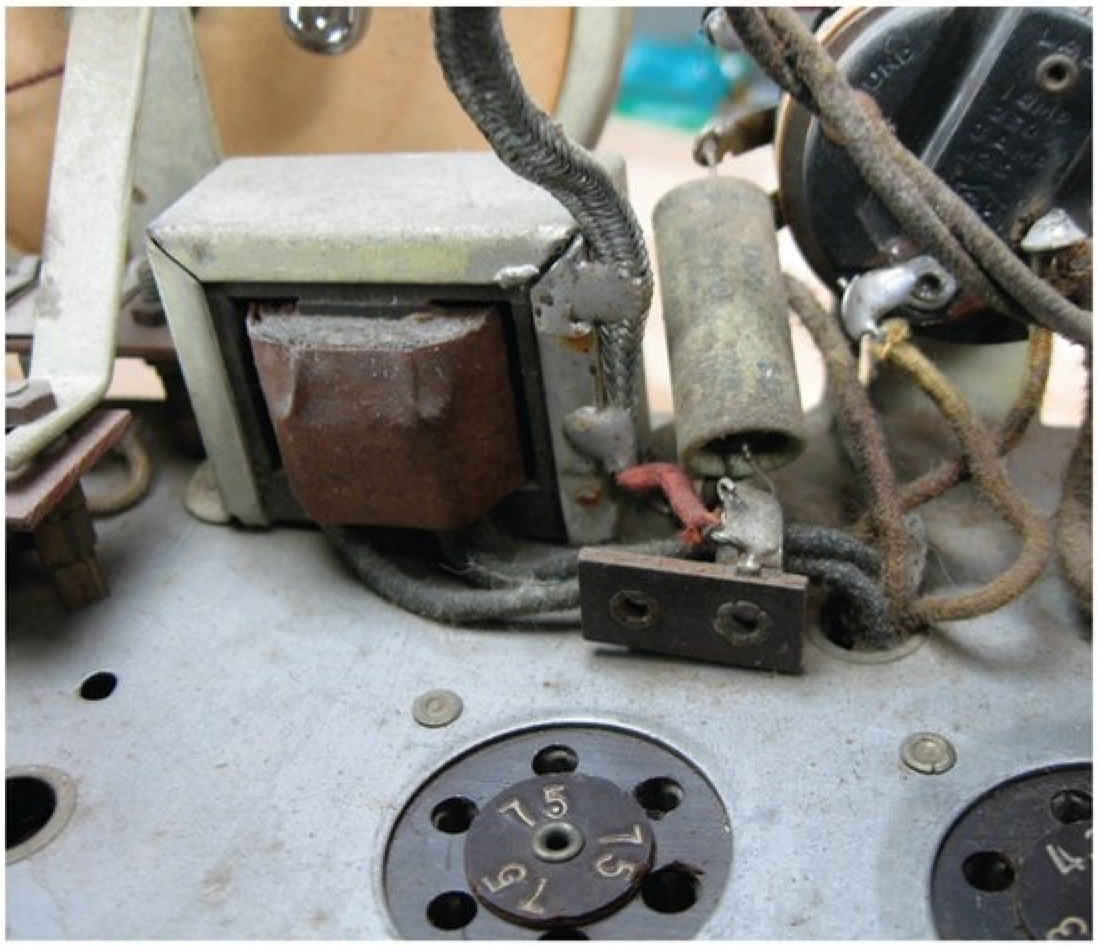
The other two oscillator coils sharing a common coil form on top of the chassis.

Back side of the radio showing components labeled.

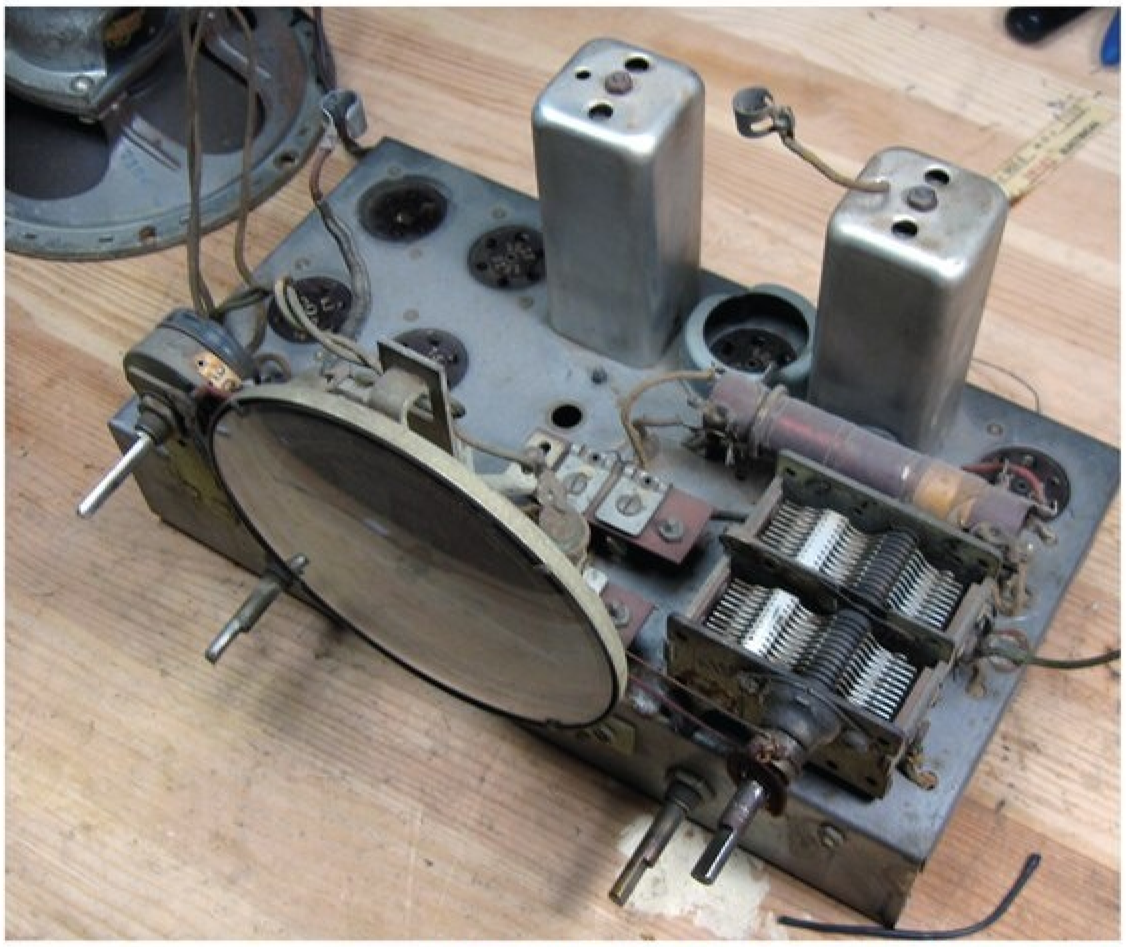




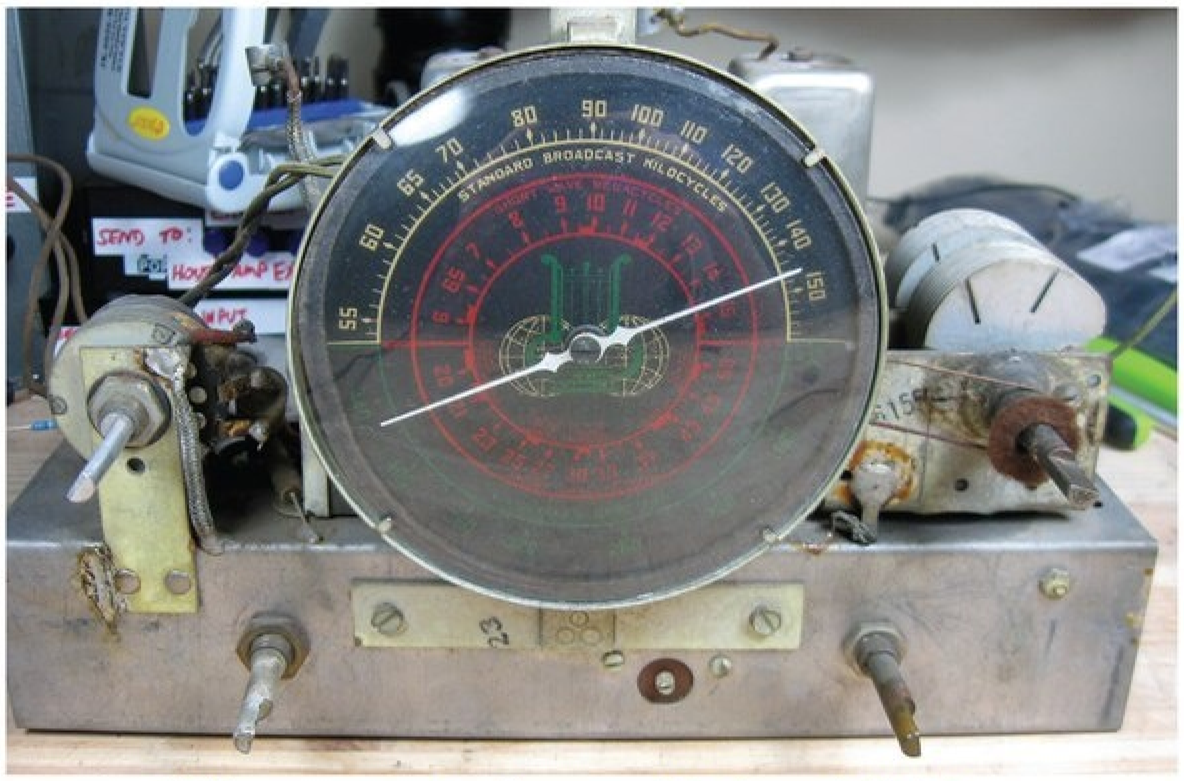
Top-down view showing other components labeled.



The filter choke.



Top of the chassis, ready for work!



What a neat looking multicolored dial face on this DeWald!

The other two components — the field coil and the output transformer — are mounted to the speaker itself.

Good permanent magnets which could be used for speakers weren't really found in consumer radios until after WW2, so this radio used an electromagnet to set up the magnetic field and energize the speaker known as the field coil. It's also used in the power supply circuit as a choke, and measuring across the field coil shows 2,309 ohms.

There's no reading given on the **schematic**, but generally 1.5-3K ohms is reasonable for a 1930's radio like this one. Finally, the output transformer is also mounted to the top, with the two sharing a common lug. The primary came in at 472 ohms DC resistance, which is pretty typical. The secondary is connected to the speaker voice coil and showed 2.1 ohms.

You might have noticed I didn't test the tubes. If you have access to a tube tester, it's fairly easy to check them out. Unless you plan to repair a lot of tube radios, a tube tester isn't a necessary investment, and you can check their function later with a few simple tests after the first power-up. If you're working on a series-string radio like this one, though, it can be useful to check out their heaters to make sure the radio will at least light up when you power it on for the first time after service.

Tube pinout diagrams can be easily found by searching for the tube's letters and the word "tube" on the Internet, like "6d6 tube." This radio uses all "old style" tubes, though, and on all old style tubes the two pins which are larger than the others and are adjacent to each other are always the heaters. If there's a measurement between those two pins, the tube heater is intact, and the heater string will light up when powered on.

Conclusion and Next Steps

These checks took me about an hour to complete, although if you're starting out on one of your first tube radios to repair, it could take quite a bit longer.

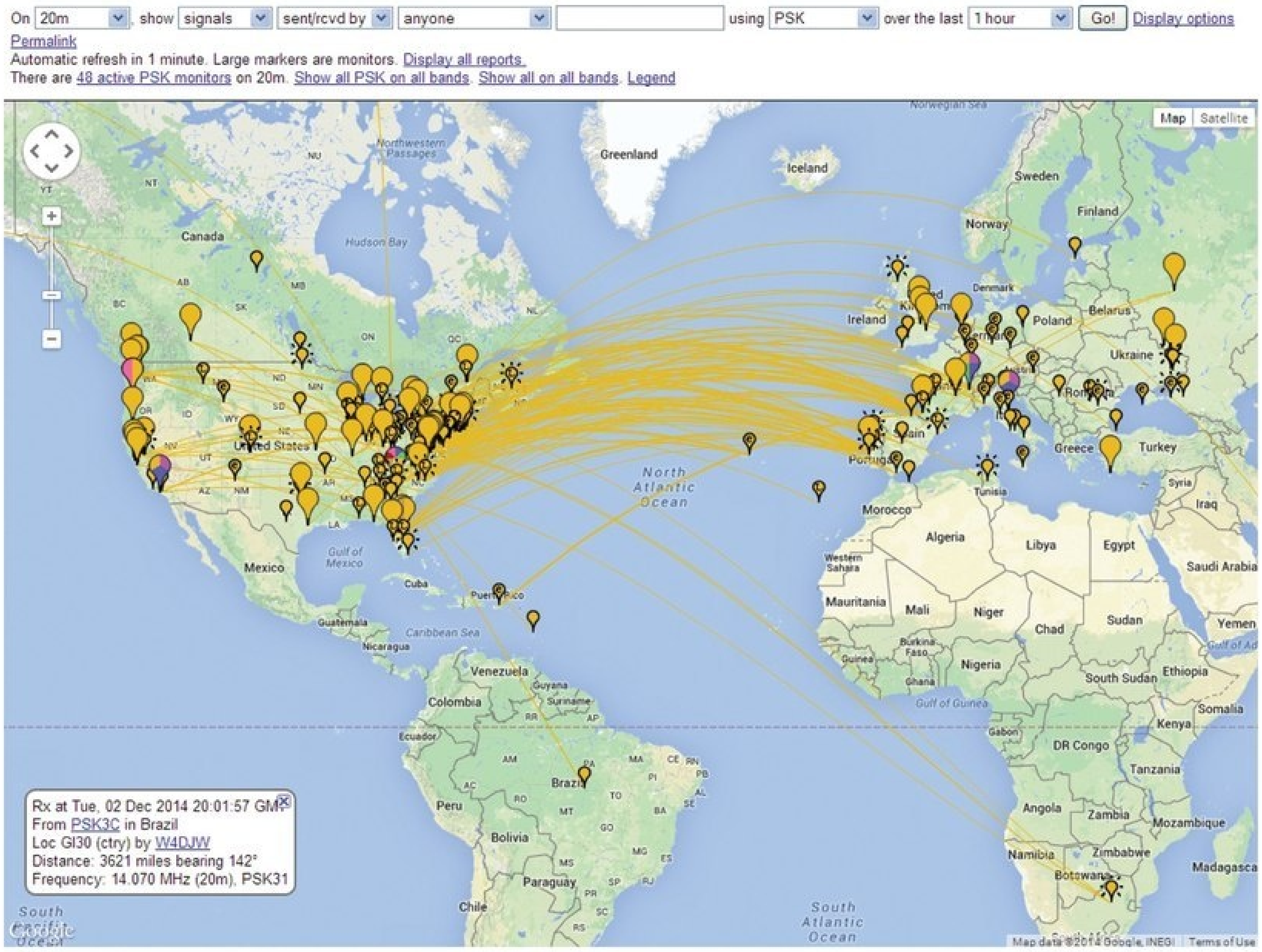
At the end of all my checks, it turns out this radio is in great shape! All the antenna, oscillator, and IF coils tested good, as did the output transformer, field coil, and filter choke. Not too bad at all! If some of those parts don't check out on your radio, not to worry. While you can't pick these parts up in your local RadioShack anymore, there are a lot of options for finding an old stock or a new functional replacement. Consult the Antique Radio Forum for some crowdsourced help if you run into trouble or need a part because someone will almost certainly have one.

Coming up, I'll be completing the under-chassis component replacement as needed, powering up for the first time to see what happens, and correcting any issues that crop up followed by an alignment. Go find yourself a vintage radio and play along! **NV**

By George R. Steber
steber@execpc.com

Receiving Data with a Low Cost Shortwave Radio

An inexpensive shortwave receiver, your computer, and some powerful software combine to make decoding many types of amateur and commercial radio data transmissions possible.



On 20m, show signals sent/rcvd by anyone using PSK over the last 1 hour Go! Display options
Permalink
Automatic refresh in 1 minute. Large markers are monitors. Display all reports.
There are 48 active PSK monitors on 20m. Show all PSK on all bands. Show all on all bands. Legend

Rx at Tue, 02 Dec 2014 20:01:57 GMT
From PSK3C in Brazil
Loc GI30 (ctry) by W4DJW
Distance: 3621 miles bearing 142°
Frequency: 14.070 MHz (20m), PSK31

FIGURE 1. PSKreporter world map.

Listening to shortwave radio is very interesting, and filled with the voices and music of far-off lands. Digging a bit deeper, there is another world of shortwave radio that does not use conventional sounds. It is the realm of data transmissions. As amazing as it seems, many of these data signals can be picked up and decoded at home using just a low cost shortwave receiver, your PC, and some specialized software. If you are connected to the Internet, your radio receptions can be reported to websites like PSKreporter.info, WSPRnet.org, and others where they will be displayed on a world map as in Figure 1. In this article, I will show you how to use a low cost ($62) shortwave radio to get you started in this exciting hobby.

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/may2015_Steber**.

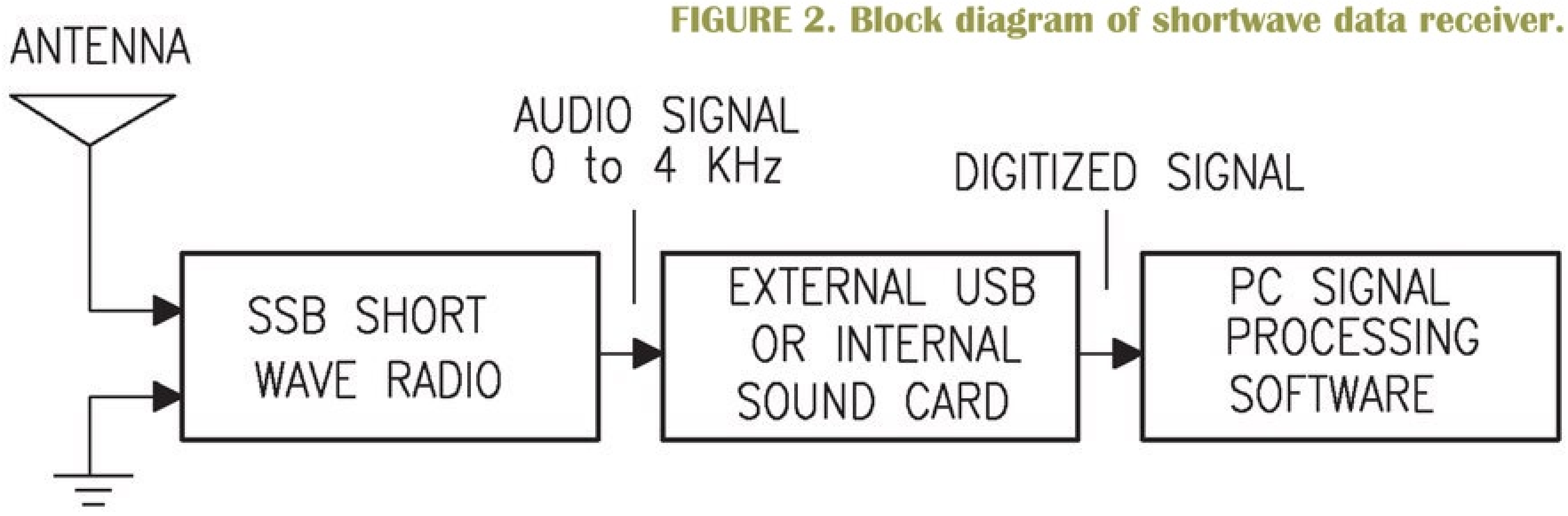


FIGURE 2. Block diagram of shortwave data receiver.

Data transmissions vary in complexity and contain a wide variety of information. Some are as simple as Morse code (CW) or Radio-teletype (RTTY), while others are more sophisticated such as AMTOR or SITOR which are similar to RTTY.

Some of the popular newer modes are Binary Phase Shift Keying (BPSK), Weak Signal Propagation Reporter (WSPR), and the weak signal digital modes JT65/9. These modes — developed by amateur radio operators — use low power and complex coding, and allow communication over great distances. Text messages are usually conveyed with these protocols. To convey images and graphics, commercial stations use radio-facsimile (FAX), while amateur stations use slow scan TV (SSTV) and EasyPal.

One of the things that all of these modes have in common is that they are *audio* modes. That is, the radio signals are first converted to audio signals using your radio receiver. Then, the sound is analyzed using audio decoders in the PC software. **Figure 2** shows how the pieces fit together.

What makes this setup work nicely is that most PCs have sound cards that can accept the audio signals from the radio. If yours does not, you can add an inexpensive USB sound card to do the job. Then, to decode the audio, it is only a matter of selecting the right software for the mode of interest.

Several software packages will be discussed to help you get started. The good news is that most of it is free and can be downloaded from the Internet.

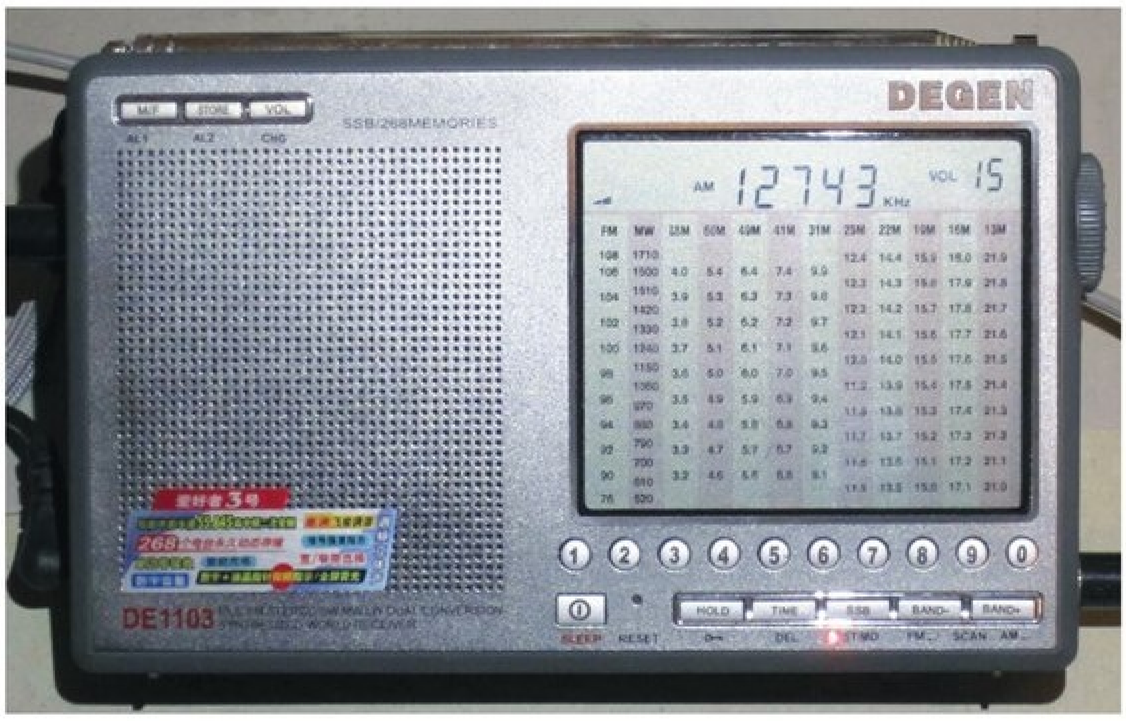


FIGURE 3. DEGEN Model 1103 SSB radio.

Selecting Your Shortwave Receiver

The basic requirements for your radio are that it be stable, have a numerical frequency readout, and receive SSB (single side band) signals. Many commercial or ham radios will do the job, but can be expensive. One popular low cost receiver that fits the bill is a DEGEN Model DE1103 as shown in **Figure 3**. It can be found on the Internet for around $62 and works well in this application.

This model stabilizes in a few minutes, has a jack for an external antenna, has digital frequency readout, and a BFO (beat frequency oscillator) for receiving SSB signals. Its small size and battery operation make it easy to take on vacations along with your laptop PC. This allows you to enjoy the hobby away from home.

Setting Up Your Shortwave Antenna

The single most important component of your setup is the antenna. Ideally, it should be a resonant antenna such as a dipole that is cut for the frequency range of interest. However, a long wire antenna placed as high as possible will work well even if it is not perfectly matched. The main thing is to get it up high and away from noise sources, like your PC.

The DE1103 has an external 3.5 mm antenna jack on the side of the case for shortwave. Unlike more costly radios that provide a 50 ohm antenna jack, the DE1103 impedance varies from 1,000 ohms at 4 MHz to 200 ohms at 14 MHz. You can use a single wire or shielded coaxial cable to connect your antenna to this jack. If you use coax, you will need to make a cable with a 3.5 mm plug on the end. To reduce noise, an antenna tuner or pre-selector can be used between the radio and the antenna.

A good ground system is very important. Try to connect your radio ground to the earth ground in your home. Failing that, use a long wire along the floor to act as a counter-poise or substitute for earth ground.

Connecting the Audio

The audio signal from your radio needs to be connected to the PC sound card line-input. If you are using an external USB sound card, use a USB cable with built-in noise suppression to prevent computer noise from getting to the radio. To connect the audio, look for the audio-out jack on the radio.

Most radios have a provision for connecting to headphones or an external speaker. For the DE1103, the connection is easy as there is a line-out jack on the side of the radio. Finally, you need to know how to adjust the line-input level in the sound card recording section of your PC.

Getting to Know Your Receiver

Spend some time getting acquainted with your receiver's operating controls. There will generally be controls to select the audio bandwidth, adjust the frequency, select the SSB mode, and adjust the BFO frequency. For the DE1130, the narrow-wide switch selects either 4 kHz or 6 kHz wide audio. In most cases, use the wide setting as powerful audio filters in the software will filter the noise. To select the desired radio frequency in kHz, enter the desired digits, press BAND+, and select SSB.

With this radio — as with many radios — you can only select the frequency to the nearest kHz since only the five most significant digits can be entered. That's okay for receiving AM stations, but for receiving SSB voice signals smaller frequency adjustments need to be made. This is done via the BFO knob on the side of the unit.

Understanding the BFO

The BFO effectively provides variable adjustment of about 1.5 kHz above and below the dial frequency as seen in **Figure 4**. It is usually un-calibrated, however. In other words, the last — or lesser — three digits of the true dial frequency setting are unknown. More frequency precision needed for data transmissions can be achieved by calibrating the BFO.

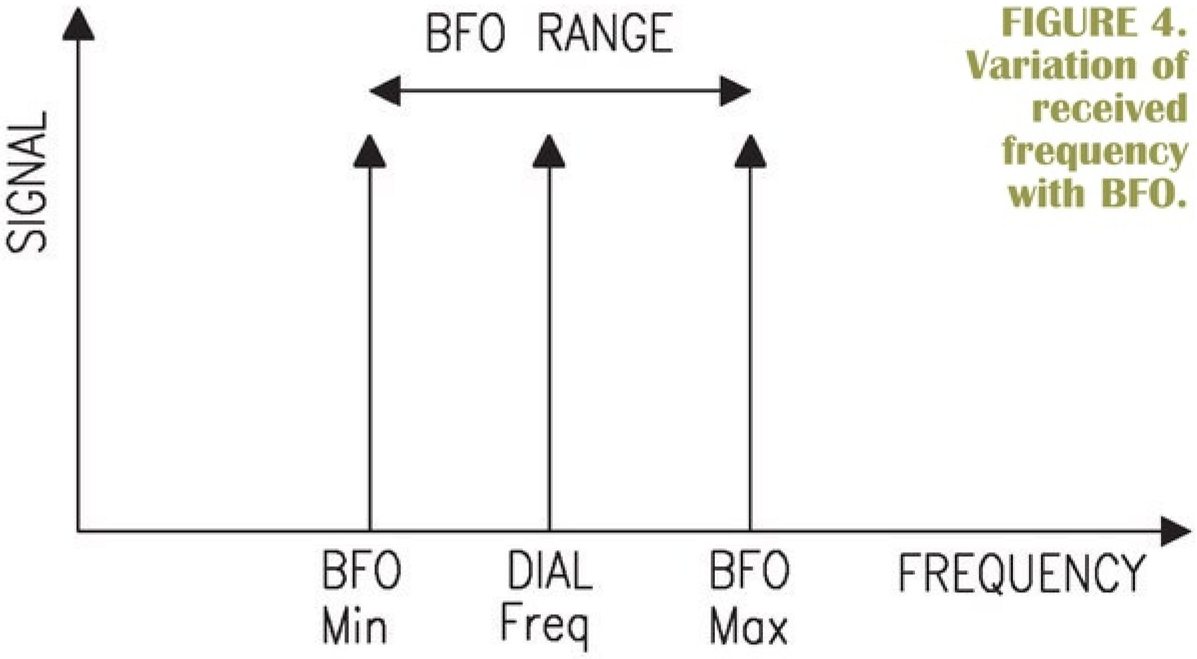


FIGURE 4. Variation of received frequency with BFO.

Typically, seven digits of frequency are needed to locate specific radio bands and for accurate signal reporting. Thus, we need to determine the true RF receiving frequency of the radio when using SSB and the BFO together.

For my DE1103, the BFO knob allows frequency adjustment of plus 1,600 Hz to minus 1,300 Hz from the displayed dial frequency. The maximum BFO frequency (1,600 Hz) is attained with the BFO knob set fully counter-clockwise (CCW). Turning the knob CW from the fully CCW position causes a decrease in the BFO frequency, gradually reducing it to 0 Hz and then further to -1,300 Hz.

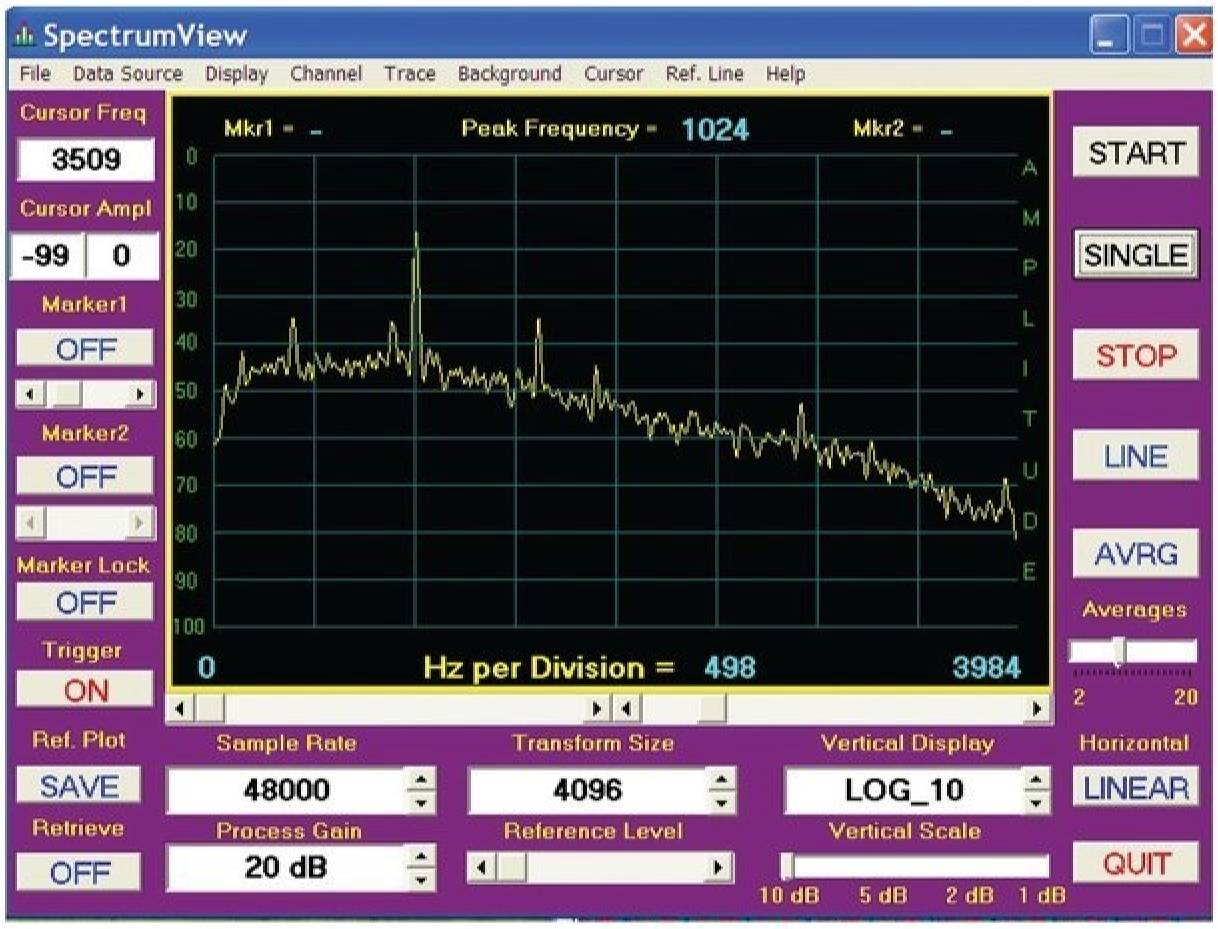
Here is an example of how the BFO affects the true radio dial. Suppose the displayed radio dial is 14,020 kHz and the BFO is set to +755 Hz. Adding the two, the true dial setting would be 14020.755 kHz (14.020755 MHz). In this example, we now have eight digits of frequency. In practice, the last digit will not be known to high accuracy. However, it will be good enough to get us to the right shortwave band segments — usually within 10 Hz.

If the BFO knob was calibrated in frequency, this would be the end of the story. Most BFOs — particularly in low cost radios like the DE1103 — are not. Another problem is that the BFO frequency will change a small amount with dial frequency. The good news is that we can easily overcome these problems using an audio spectrum analyzer on our PC and two standard candle radio stations separated widely in frequency.

These standard candle radio stations can be an AM radio station near 1,000 kHz and a shortwave station like WWV at 10 MHz. WWV has a very accurate frequency, and AM stations are required to be within 20 Hz of their assigned frequency. This allows us to perform a calibration for our radio's BFO. It is easy and fun. Here's how to do it.

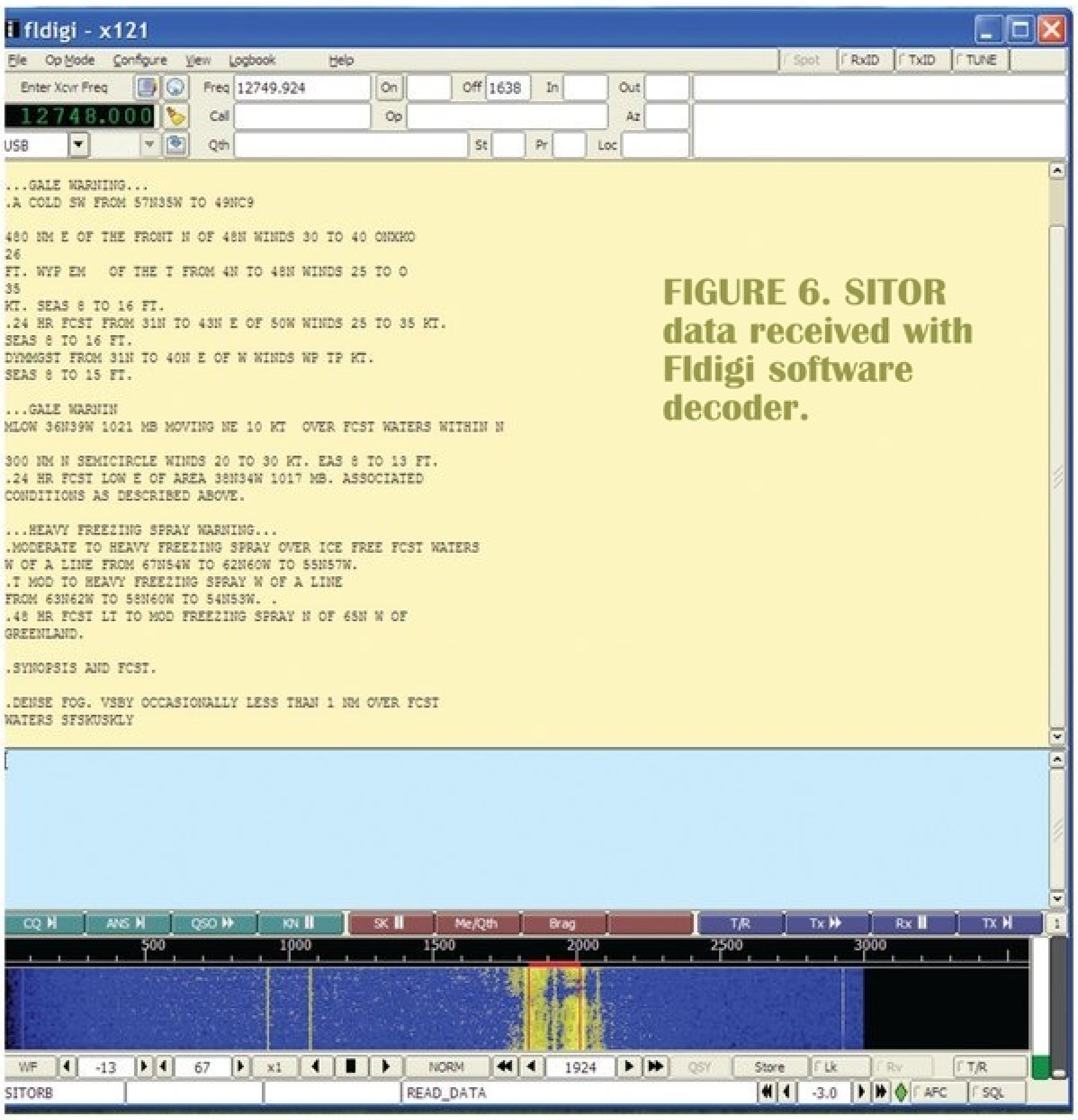
Calibrating Your BFO on SSB

First, get an audio spectrum analyzer (SA) running on your PC. A good choice is the free program



Cursor Freq 3509
Cursor Ampl -99 0
Marker1 OFF
Marker2 OFF
Marker Lock OFF
Trigger ON
Mkr1 = -
Peak Frequency = 1024
Mkr2 = -
0
Hz per Division = 498
3984
Averages
Ref. Plot SAVE
Retrieve OFF
Sample Rate 48000
Process Gain 20 dB
Transform Size 4096
Reference Level
Vertical Display LOG_10
Vertical Scale
Horizontal LINEAR

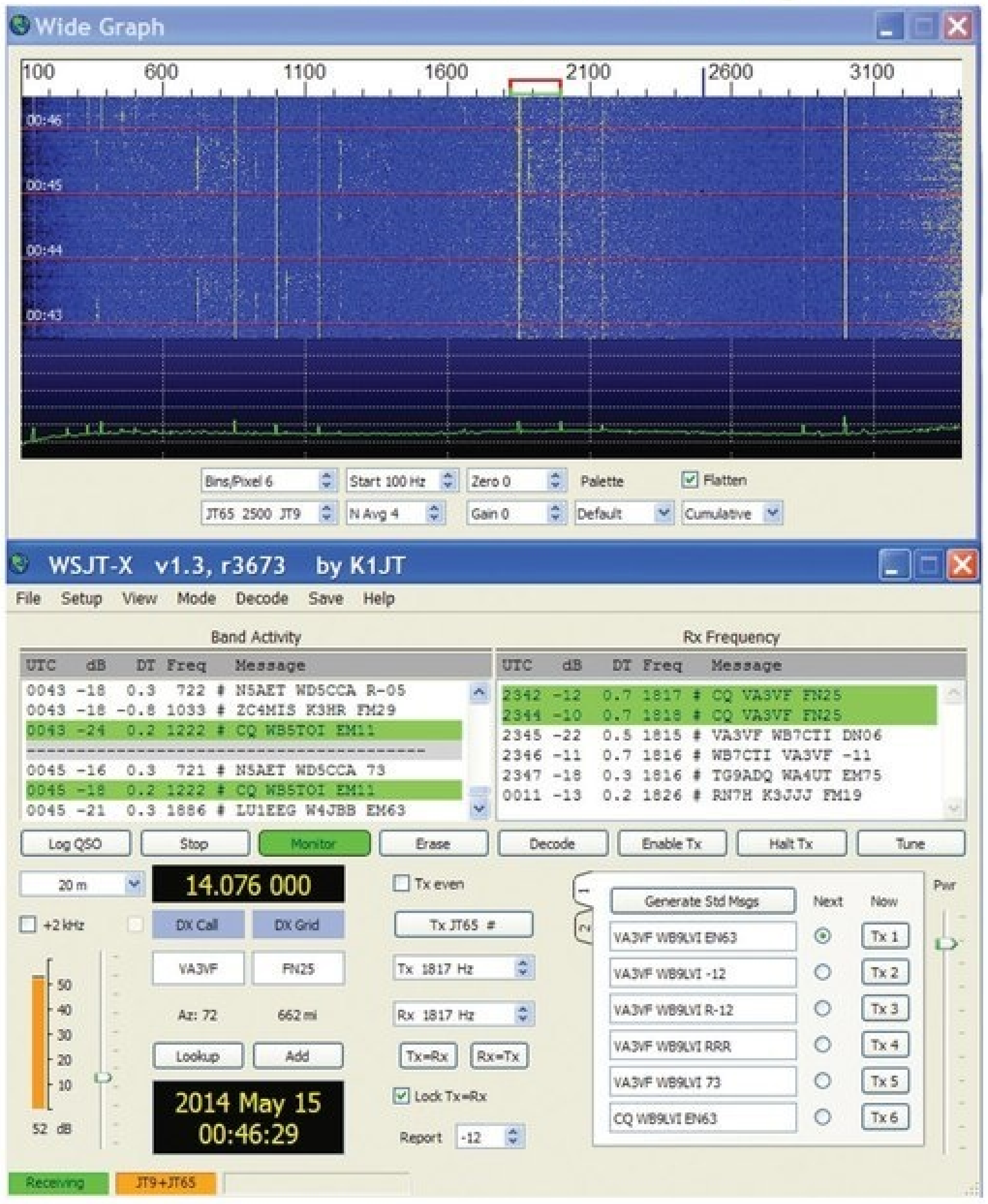
FIGURE 5. SpectrumView audio spectrum analyzer.



```
...GALE WARNING...
.A COLD SW FROM 57N35W TO 49NC9

480 NM E OF THE FRONT N OF 48N WINDS 30 TO 40 ONKKO
26
FT. WYP EM   OF THE T FROM 4N TO 48N WINDS 25 TO O
35
KT. SEAS 8 TO 16 FT.
.24 HR FCST FROM 31N TO 43N E OF 50W WINDS 25 TO 35 KT.
SEAS 8 TO 16 FT.
DYMMGST FROM 31N TO 40N E OF W WINDS WP TP KT.
SEAS 8 TO 15 FT.

...GALE WARNIN
MLOW 36N39W 1021 MB MOVING NE 10 KT  OVER FCST WATERS WITHIN N

300 NM N SEMICIRCLE WINDS 20 TO 30 KT. EAS 8 TO 13 FT.
.24 HR FCST LOW E OF AREA 38N34W 1017 MB. ASSOCIATED
CONDITIONS AS DESCRIBED ABOVE.

...HEAVY FREEZING SPRAY WARNING...
.MODERATE TO HEAVY FREEZING SPRAY OVER ICE FREE FCST WATERS
W OF A LINE FROM 67N54W TO 62N60W TO 55N57W.
.T MOD TO HEAVY FREEZING SPRAY W OF A LINE
FROM 63N62W TO 58N60W TO 54N53W. .
.48 HR FCST LT TO MOD FREEZING SPRAY N OF 65N W OF
GREENLAND.

.SYNOPSIS AND FCST.

.DENSE FOG. VSBY OCCASIONALLY LESS THAN 1 NM OVER FCST
WATERS SFSKUSKLY
```

FIGURE 6. SITOR data received with Fldigi software decoder.

FIGURE 7. JT65 decoding windows.



Band Activity

```
UTC   dB   DT Freq   Message
0043 -18  0.3  722 # N5AET WD5CCA R-05
0043 -18 -0.8 1033 # ZC4MIS K3HR FM29
0043 -24  0.2 1222 # CQ WB5TOI EM11
---------------------------------------
0045 -16  0.3  721 # N5AET WD5CCA 73
0045 -18  0.2 1222 # CQ WB5TOI EM11
0045 -21  0.3 1886 # LU1EEG W4JBB EM63
```

Rx Frequency

```
UTC   dB   DT Freq   Message
2342 -12  0.7 1817 # CQ VA3VF FN25
2344 -10  0.7 1818 # CQ VA3VF FN25
2345 -22  0.5 1815 # VA3VF WB7CTI DN06
2346 -11  0.7 1816 # WB7CTI VA3VF -11
2347 -18  0.3 1816 # TG9ADQ WA4UT EM75
0011 -13  0.2 1826 # RN7H K3JJJ FM19
```

VA3VF WB9LVI EN63	●	Tx 1
VA3VF WB9LVI -12	○	Tx 2
VA3VF WB9LVI R-12	○	Tx 3
VA3VF WB9LVI RRR	○	Tx 4
VA3VF WB9LVI 73	○	Tx 5
CQ WB9LVI EN63	○	Tx 6

SpectrumView from WD6CNF. It will be assumed here that your radio audio output is connected to the sound card on your computer and SpectrumView is running. Allow the radio to warm up to reduce drift. Remember there are two things we want to find: the BFO frequency and how it changes with frequency.

It's easy to determine the BFO frequency. Tune in a local AM station of known frequency and select SSB on the receiver. There will be an audio tone and a peak in the audio spectrum as shown in **Figure 5.** This peak is due to the carrier of the AM station. Adjust the BFO knob back and forth, and watch the tone peak move left or right. The peak frequency is shown at the top of display. This tone peak is the BFO frequency.

Okay, now we want to find out how the BFO changes with RF dial frequency. We can do this by tuning in another station far away in frequency from the AM station and noting the change in BFO frequency. A good choice is WWV at 10 MHz as previously mentioned. (In a pinch, an RF signal generator can be used in place of WWV.)

Let's say that we tuned to a local AM station at 1.130 MHz and set the BFO to 970 Hz by observing SpectrumView. Then, say, we tuned to WWV at 10 MHz and found the BFO peak had changed to 1,024 Hz. This shows that the BFO increased 54 Hz over the 8.87 MHz frequency difference between the two stations. This is a rate of 6.08 Hz per MHz. It is a small change, but worth correcting for in the interest of accuracy. (Your AM station frequency and numbers, of course, will be different.) *Remember this rate value.* It is all we need to set the BFO to any value at any frequency.

Here is an example of how to do this.

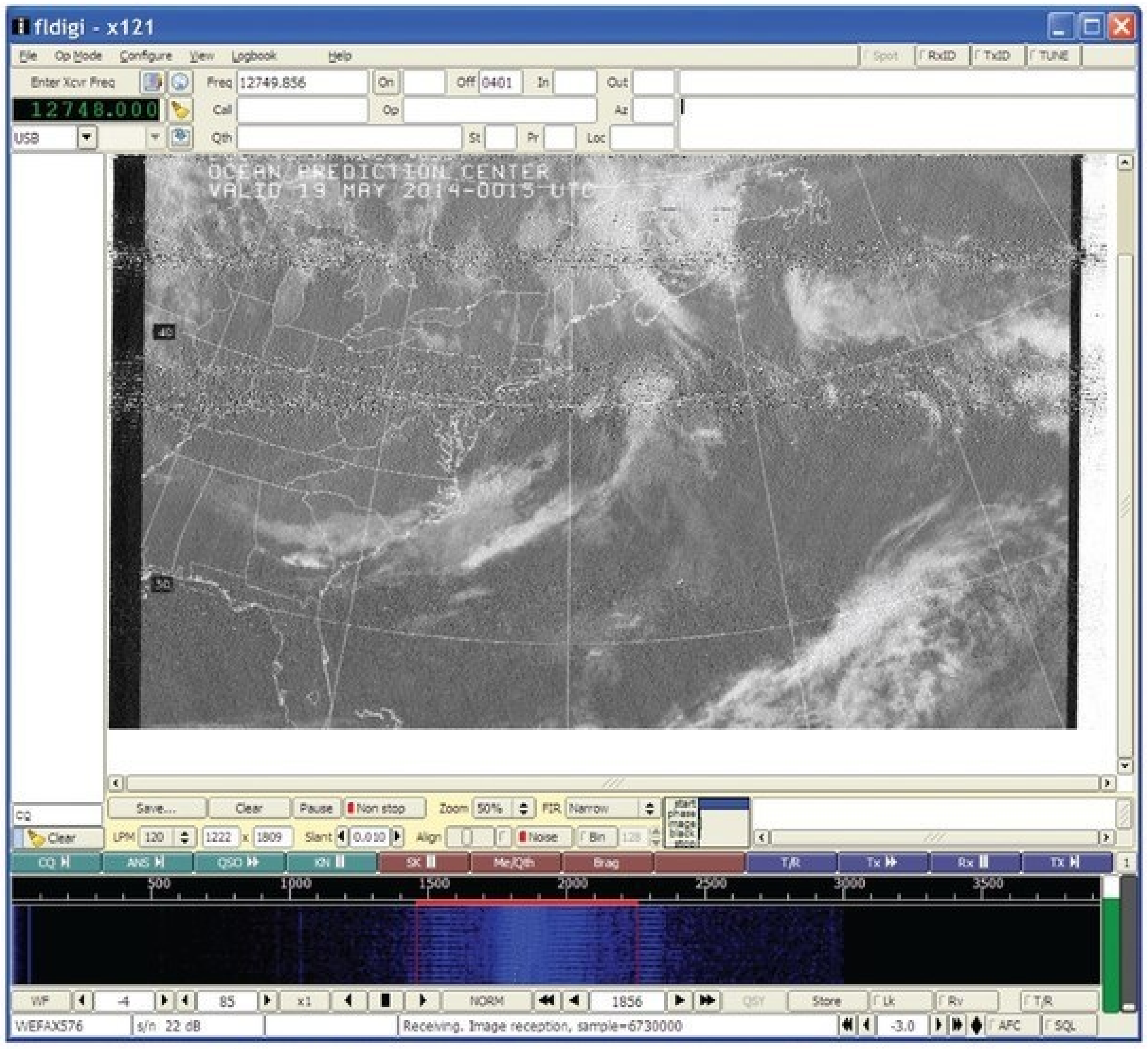
Suppose we wish to set the BFO to 600 Hz at 14.095 MHz. (Note that this corresponds to the true dial of 14.095600 MHz in SSB mode.) We will use the standard candle AM station at 1.130 MHz. Note the frequency difference (14.095 – 1.130) = 12.965 MHz. Hence, the BFO frequency at 1.13 MHz would correspond to 600 – 6.08 x 12.965 = 521.1 Hz.

At this point, tune in the AM station at 1.130 MHz in SSB mode and adjust the BFO knob for 521 Hz. Now, when you set 14.095 on the radio dial, it will correspond to BFO of 600 Hz and the *true* dial frequency of 14.095600 MHz. In this case, the AM station is our standard candle, and since it is a local station it is always available for setting the BFO using SpectrumView. Station WWV is no longer needed.

Note that the BFO setting can be a negative frequency. Suppose you want a BFO setting of 0 Hz at 14.076 MHz which is a true dial frequency of 14.076000 MHz. Similar to above, we have 0 – 6.08 x 12.946 = -78 Hz. For this case, just rotate the BFO knob past 0 Hz to negative frequencies, as monitored by SpectrumView at the standard candle of 1.13 MHz.

Once the BFO is set, do not adjust it except to change to another effective dial setting. It may drift a little with temperature, so check it periodically. Remember that SSB mode must be selected to use it. BFO calibration is very useful for selecting the starting radio frequency segments for BPSK, WSPR, and JT65, which usually need an accurate true dial setting.

Since many of the signals you will be receiving are barely audible, this procedure puts you on the right frequency.



12748.000

USB

Freq 12749.856

Off 0401

Zoom 50%

FIR Narrow

LPM 120

1222 x 1809

Slant 0.010

WF -4 85

1856

WEFAX576

s/n 22 dB

Receiving. Image reception, sample=6730000

-3.0

FIGURE 8. Weather FAX image received with Fldigi.

Grid Squares

A geographic coordinate system used by amateur radio operators and others around the world is the Maidenhead Locator System (MLS). It is also commonly referred to as grid locators or grid squares. The MLS compresses latitude and longitude into a string of characters to allow position information to be transmitted with limited precision. This is very useful when determining distances and angles of radio transmissions.

Many web logging sites such as PSKreporter and WSPR use grid squares. Most data decoding programs have means to automatically send grid squares, as well as frequency information, call sign, or name directly from the program. So, if you are connected to the Internet and want to provide signal reports, make sure to enable that feature in the program, and have your specific grid square and other information entered properly. To find your specific grid square, go to **www.levinecentral.com/ham/grid_square.php** and enter your zip code or city. For example, Chicago has the grid square EN61ev.

CW, RTTY, and BPSK

In my opinion, the best program for receiving CW, RTTY, and BPSK is Fldigi (Fast and Light Digital Modem Program). **Figure 6** shows Fldigi in action receiving SITORB. It is a free program developed by radio amateurs. It handles many other modes as well, and has numerous features including a built-in spectrum display and audio filters. You can also enter your frequency, call sign, or name, and enable automatic reporting to PSKreporter.

Fldigi can be downloaded from **www.w1hkj.com/Fldigi.html**. Check the Internet for frequencies where these modes can be found. For example, PSK-31 can be found at 14.070000 MHz and SITOR at 12.577000 MHz.

WSPR, JT9, and JT65

These modes are used for weak signal communication. Oftentimes, these signals are so weak they are not perceptible by ear. These modes are used in special segments of the ham bands. For example, WSPR can be found on 14.095600 MHz, and JT65 can be found on 14.076000 MHz. Note the precision of the dial settings; see **Figure 7** for a JT65 screenshot.

Included with the programs are dial frequencies for all bands. WSPR has its own website for reporting spots, while JT9/JT65 uses PSKreporter. These programs can be found at **www.physics.princeton.edu/pulsar/K1JT**.

These weak signal modes require coordination of your PC with Universal Coordinated Time (UTC) to within one or two seconds. So, make sure to set your computer clock accordingly, and check it often.

Software for FAX, SSTV, and EasyPal

FAX can also be received with Fldigi. An example of a noisy weather fax is shown in **Figure 8**. Fax signals can often be found around 12.748 and 12.788 MHz. Slow scan TV signals can be found around 14.230 MHz. Several programs are available on the Internet for SSTV.

One of the newer picture modes is EasyPal, which transmits pictures digitally. It can be found at 14.233 MHz. Perfect pictures can be received if the signal manages to make it through the noise. Search for it on the Internet.

Final Comments

Hopefully, you have learned a little bit about how to receive and decode shortwave data signals. It is a very interesting hobby, and new modes are cropping up all the time. If you find the subject of decoding radio signals fascinating, you may want to consider becoming a radio amateur. Hams are involved with building and studying receivers, transmitters, and antennas, and experimenting with the radio modes WSPR, JT65, and BPSK discussed here.

If you are considering joining the fraternity of radio amateurs, the ARRL website is the place to start. In the meantime, have fun with your shortwave radio decoder. **NV**

Serial I/O Data Interfaces: Part 2

Gigabit interfaces make serial I/O practical.

Back in the January 2015 issue, I summarized all the popular low speed serial interfaces like RS-232, I²C, CAN, and others. In this article, I will familiarize you with the high speed gigabit serial interfaces that dominate I/O today. When you think about it, what electronic product does not have a serial interface? I can't think of one. We all use at least one of these interfaces regularly.

Parallel data transfers are generally faster than serial, but are more expensive as they require more hardware, larger connectors, and one wire per bit. Furthermore, as parallel wires get longer, the inter-wire capacitance and the inductance of the wires and crosstalk greatly impact the data rate, so shorter connections are necessary to get any speed. Data skew on the lines is also a problem. Parallel connections are not very useful at rates beyond about 100 Mb/s, and that can be achieved only over a foot or so. Serial transfers are less expensive, simpler, and can achieve high speeds over longer distances. Today, serial interfaces dominate.

There are literally dozens of gigabit serial interfaces in use, but I will only summarize the most popular ones here like USB, HDMI, PCIe, LVDS, Ethernet, and OTN.

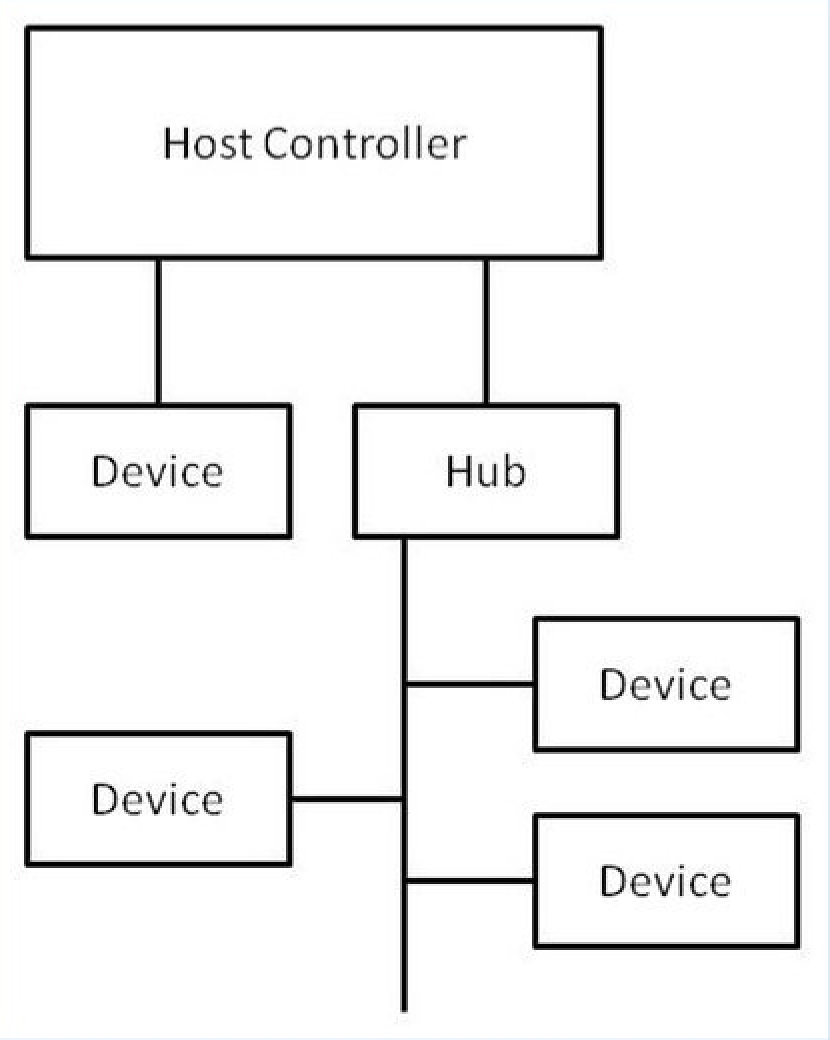
USB

By far, the most widely used gigabit serial other than Ethernet is the Universal Serial Bus (USB). It is used on all PC peripherals like keyboards, mice, printers, disk drives, Flash memory sticks, and others. This interface has been around for quite a while, and has been upgraded steadily over the years to improve its data rate. The original version was slow at 12 Mb/s, but it was faster than the RS-232 and parallel printer ports it replaced. Version 2.0 boosted data rates to 480 Mb/s which is fast enough for most peripherals.

The basic USB configuration is shown in **Figure 1**. The host — usually a PC or laptop — controls all operations and allows multiple slave nodes to connect. The network topology is a star, with one port per slave. If you need more than the one or two ports normally supplied on a PC or laptop, you can add a hub. A hub connects to one of the star nodes and offers multiple bus connections. Most systems can support up to 127 nodes. Communications are half duplex which means two-way, but only one direction at a time. Data is sent in packets with up to 1,024 bytes per packet.

The interface is really simple with just four wires and connector pins. It only uses a single twisted pair for data. The signals are differential which helps with noise mitigation. The only other connections are +5V and ground that are used to power external peripherals if needed. The maximum cable length is five meters for versions 1.1 and 2.0 to get the higher data rate. Repeaters and extenders are available to lengthen this if needed.

More recent versions 3.0 and 3.1 use a larger, more complex 10-pin connector with three differential data pairs. This permits full duplex operation or simultaneous send-receive. Data rates are bumped to 5 Gb/s and 10 Gb/s. This restricts cable length to three meters. This faster version finds applications in video equipment.



■ FIGURE 1. Topology of a USB connection. The host is usually a PC or laptop. The hub serves as an expander.

Go to **www.nutsvolts.com/index.php?/magazine/article/May2015_OpenCommunication** to comment on this article.

HDMI

Another widely used gigabit interface is the High-Definition Multimedia Interface (HDMI). It was designed for and is used primarily for video data transfer. It is found in HDTV sets, DVRs, Blu-Ray players, set top boxes, video games, camcorders, digital video projectors, and even some PCs and laptops.

HDMI is designed to carry uncompressed video and either compressed or uncompressed audio. The general configuration is shown in **Figure 2**. With today's high pixel count, long color words, and fast refresh rates, speeds must be fast to keep up with the video action. The first versions of HDMI could handle 1.65 Gb/s. Later versions bumped this up to 3.4 Gb/s. The latest version 2.0 delivers 6 Gb/s — enough to support the latest 4K ultra HD video.

As for cables and connectors, cable length is restricted to five meters to achieve those elevated data rates. With special cable, lengths of up to 15 meters are possible. You can also buy repeaters and extenders to string that out for as many meters as needed.

Connectors are another matter. The standard has 19 pins. A larger extended version has 29 pins. Mini and micro 19-pin connectors are also available for some devices.

Interface technology for HDMI is pretty simple. It is strictly simplex; that is, one way only from the video source to the destination. However, multiple channels are used. (Refer again to **Figure 2**.)

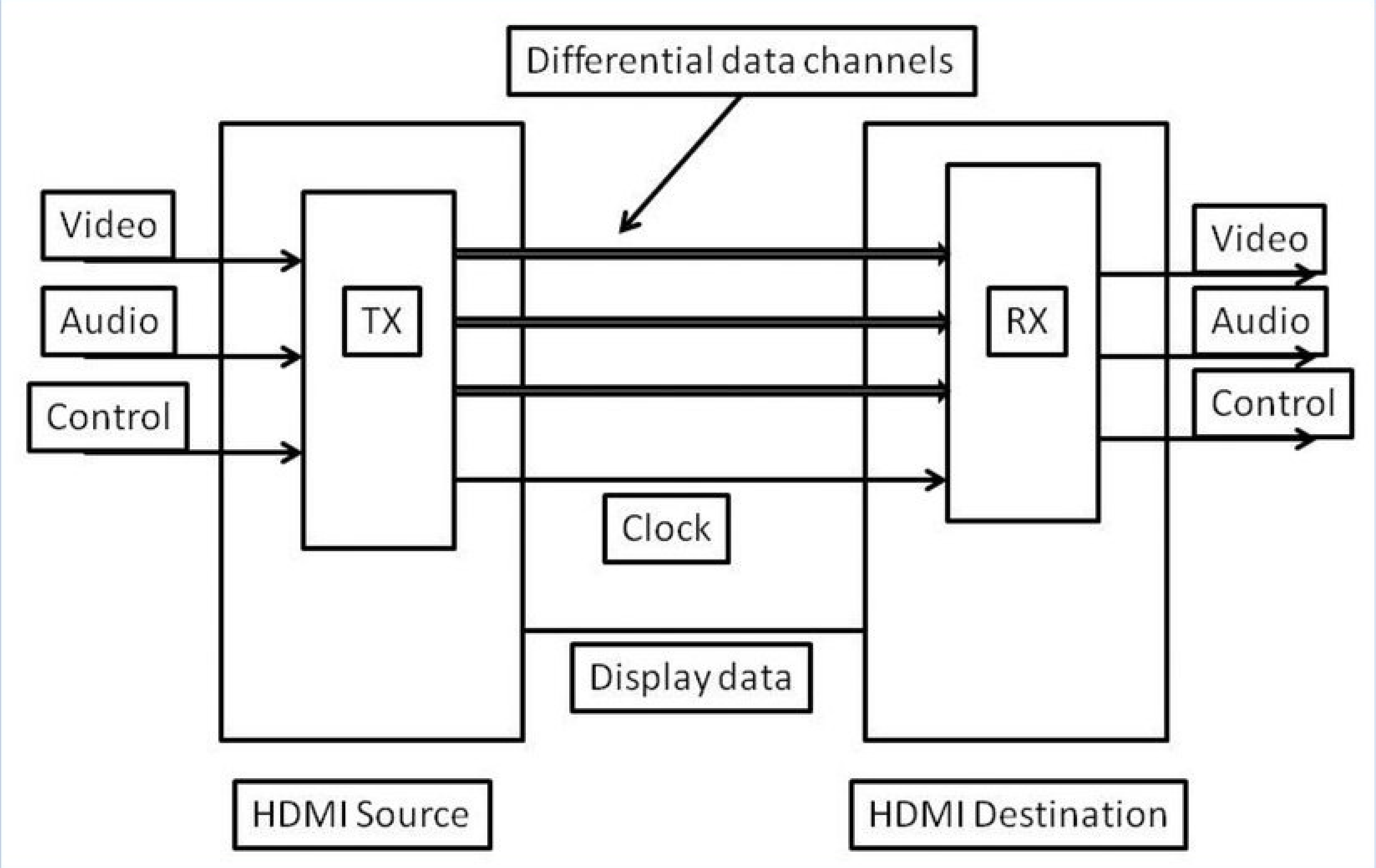
There are three differential data channels for video data and control information. These channels use what is called time minimized differential signaling (TMDS) to time-multiplex the video, audio, and control information. The interface even incorporates a standard I2C interface to handle data for the type of display. The interface also includes the high bandwidth digital content protection encryption software to prevent the recording of proprietary video.

PCIE

PCI Express is the serial data interface that replaced the PCI and PCI-X parallel interfaces in PCs and laptops. It is the interface on the motherboards and backplanes in computers that interconnect the processors, co-processors, memory, and peripheral chips. The interface is also implemented on connectors for attachment to expansion cards.

What PCIe does is replace those 32-bit parallel busses with one or more serial data channels. Each channel is a pair of differential lines, so that full duplex/simultaneous transmit and receive operations are possible. Each of these channels can transmit at gigabit speeds depending on the version. Version 1 channels run at 2.5 Gb/s, version 2 at 5 Gb/s, version 3 at 8 Gb/s, and version 4 at 16 Gb/s. If you need a greater data rate, you just use four, eight, or 16 of these channels in parallel. Data bytes are interleaved between channels. For example, with four channels (x4) at 5 Gb/s, you get a total transfer rate of 20 Gb/s. That translates to two gigabytes per second, or 2 GB/s since 8B/10B encoding is used. The maximum rate with sixteen 16 Gb/s channels is 256 Gb/s.

At these speeds, range is short; typically inches on a PC bus that runs on a motherboard. Short cables are available for testing. Data is transferred in packets using a standard protocol which includes a 32-bit CRC for error detection and correction. Most connections are just point-to-point, but some applications use a fast PCIe switch IC that lets one CPU serve multiple peripherals.



■ **FIGURE 2.** The HDMI source may be a cable or satellite TV set top box, while the destination is an HDTV set.

LVDS

One of the interfaces you will hear about is Low Voltage Differential Signaling, or LVDS. This is just the physical layer (connectors, cable medium, logic levels, etc.) of an interface. It does not define a protocol for data transfer. It is the physical layer of many other interfaces such as PCIe (just described).

An LVDS interface uses differential pairs for point-to-point connections. The pairs have a transmission line impedance of 100 ohms. Differential logic levels are typically ± 350 mV. The M-LVDS version of the standard also supports a multidrop bus connection format with up to 32 drops. Typical data rates are in the 1 Gb/s to 3 Gb/s range. The length of connections is typically inches on a PCB (printed circuit board). A special M-LVDS version uses ± 480 mV levels, and can be used at distances up to 10 meters or more at lower data rates.

Physical Layer (medium)	Range (meters) Up to…	40GE	100GE
Backplane	1 m	40GBASE-KR4	100GBASE-KP4
Improved backplane	1 m		100GBASE-KR4
Twinax copper coax cable	7 m	40GBASE-CR4	100GBASE-CR10
CAT8 twisted pair	30 m	40GBASE-T	
MMF (OM3)	100 m	40GBASE-SR4	100GBASE-SR10
MMF (OM4)	125 m	40GBASE-SR4	100GBASE-SR10
SMF	2 km	40GBASE-FR	
SMF	10 km	40GBASE-LR4	100GBASE-LR4
SMF	40 km		100GBASE-ER4

■ **Table 1** 40G and 100G versions of 802.3ba showing medium and range.

ETHERNET

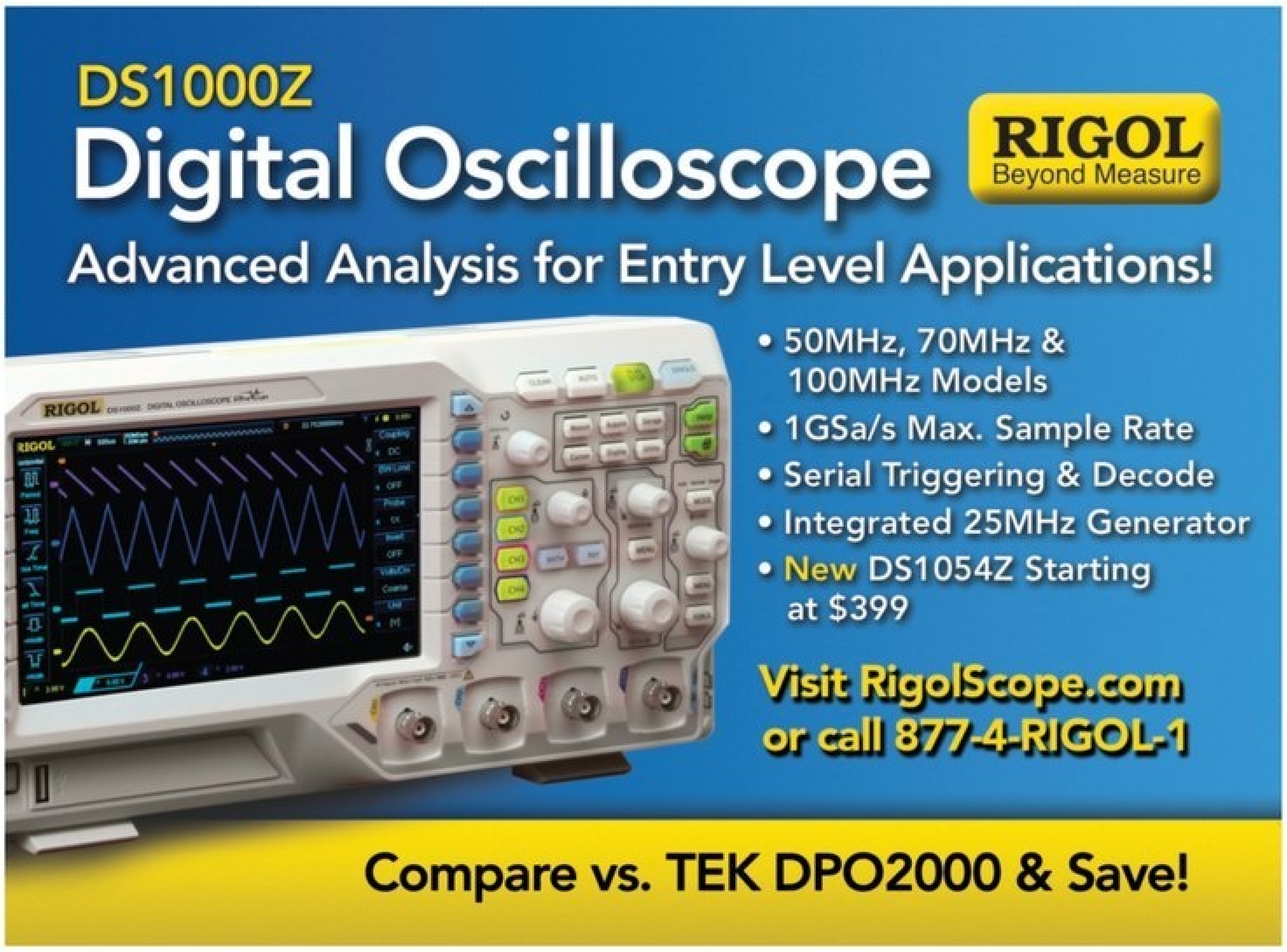
What can I say about Ethernet? It is just everywhere. It is the overall networking technology of choice. It was originally developed as a local area network (LAN), but is now used in larger and longer range metropolitan area networks (MAN) and even wide area networks (WAN) for Internet connectivity. Other versions are used on backplanes in servers and other communications gear.

The original versions were not gigabit interfaces. The main standard IEEE 802.3 started at 10 Mb/s, but soon jumped to 100 Mb/s. The 10/100 versions were widely implemented and many such LANs are still around. Soon, new standards emerged for 1 Gb/s and 10 Gb/s. Today, we have 40 Gb/s and 100 Gb/s versions.

Ethernet started out with several versions that use coax cable. Those were soon replaced with cheaper and more flexible unshielded twisted pair (UTP) cables such as the popular CAT5/5e/6/7 varieties. Several fiber optic cable versions are also defined and widely used — especially with the higher data rates. **Table 1** gives you a glimpse of the Gigabit Ethernet (GE) 40 and 100 Gb/s versions.

The 40 and 100 Gb/s rates are achieved in several ways. To get 40 Gb/s, you can use multimode fiber (MMF) to get a range of 100 to 125 meters, or a single mode fiber (SMF) to achieve ranges up to 10 km. Over shorter distances, four 10 Gb/s data streams on four parallel cables can give you an aggregate rate of 40 Gb/s

To get 100 Gb/s, you can use special optical modulation techniques on SMF or MMF to reach ranges of 100 meters to 40 km. In some formats, 100 Gb/s is obtained with 10 parallel 10 Gb/s cables or four 25



Gb/s cables. Some versions multiplex four or 10 data streams on different wavelengths of infrared light on a single cable.

Today, most PCs, laptops, and other common devices have a 1 Gb/s port. The higher data rate versions are used in servers, routers, switches, and other data center gear. The network topology is a physical star, but a logical bus. Hubs and switches connect multiple nodes to the network and related servers.

Ethernet continues its expansion. A 25 Gb/s version to connect Wi-Fi access points in hot spots is under development. Also being created is a set of standards for 400 Gb/s. Yes, 400 Gb/s that will be used mostly over shorter distances in data centers to connect servers, routers, and switches. A one terabit/1 Tb/s version is undoubtedly in the future.

OTN

Have you heard of this? Optical Transport Network is the long haul fiber optic technology used in upgrading the Internet backbone. It is replacing proprietary links, as well as the entrenched Synchronous Optical Network (SONET) that has been around for decades. SONET ran out of speed with its peak 40 Gb/s version. It was used in mostly rings, as well as direct point-to-point links. OTN provides higher speeds and other advantages.

OTN defines four basic data rates: OTU1 at 2.66 Gb/s; OTU2 at 10.7 or 11.09 Gb/s; OTU3 at 43.01 Gb/s; and OUT4 at 112 Gb/s. These rates allow OTN to carry other network technologies. SONET is a synchronous network that runs in step with a clock, and all operations are timed. Everything must be in sync with the clock. OTN is an asynchronous technology, and data is carried in large packets or frames. The data inside can be other network protocols like SONET or Ethernet. That is why OTN is called the digital "wrapper" since it can transport almost any other data format at Internet speeds.

OTHER GIGABIT INTERFACES

There are another dozen or so gigabit interfaces. Here are a few that you may have heard of: DisplayPort, Fibre Channel, HyperTransport, Infiniband, MIPI, RapidIO, Lightening, SAS, SATA, and Thunderbolt. I probably missed some, but you get the idea. Serial I/O is fast and economical, and better fits digital data transmission today than parallel — now that the technology is there to deliver it. **NV**

ELECTRONET

For the ElectroNet online, go to **www.nutsvolts.com** click ***Electro-Net***

Did You Know **Preferred Subscribers get access to all the digital back issues of *Nuts & Volts* for free?**

Call for details

1-877-525-2539

■ BY FRED EADY

Wi-Fi on the Big Wire

To quote a famous television Marine, "Surprise, surprise, surprise!" Once I figured out what embedded USB was all about, I included both host and downstream USB interfaces in most all of my projects. Years ago, I was toying with Bluetooth and poo-pooed it as a fad that wouldn't catch on. These days, I'm doing live online Bluetooth lectures, Bluetooth columns, and Bluetooth-On-Your-Phone projects. Wonders never cease.

I was about to poo-poo Wi-Fi modules with embedded TCP/IP stacks as I've never found one that could act as both server and client in a deep TCP/IP way. In recent editions of Design Cycle, we've examined the new ACKme Moray development board and I wanted to make sure we had covered all of its important aspects. I decided to follow the Moray configuration path that seemed to lead to an HTTP server function. To my delight, I discovered that the Moray development board we previously discussed can indeed serve web pages in a most serious way, using an API-based system called WiConnect.

What is WiConnect

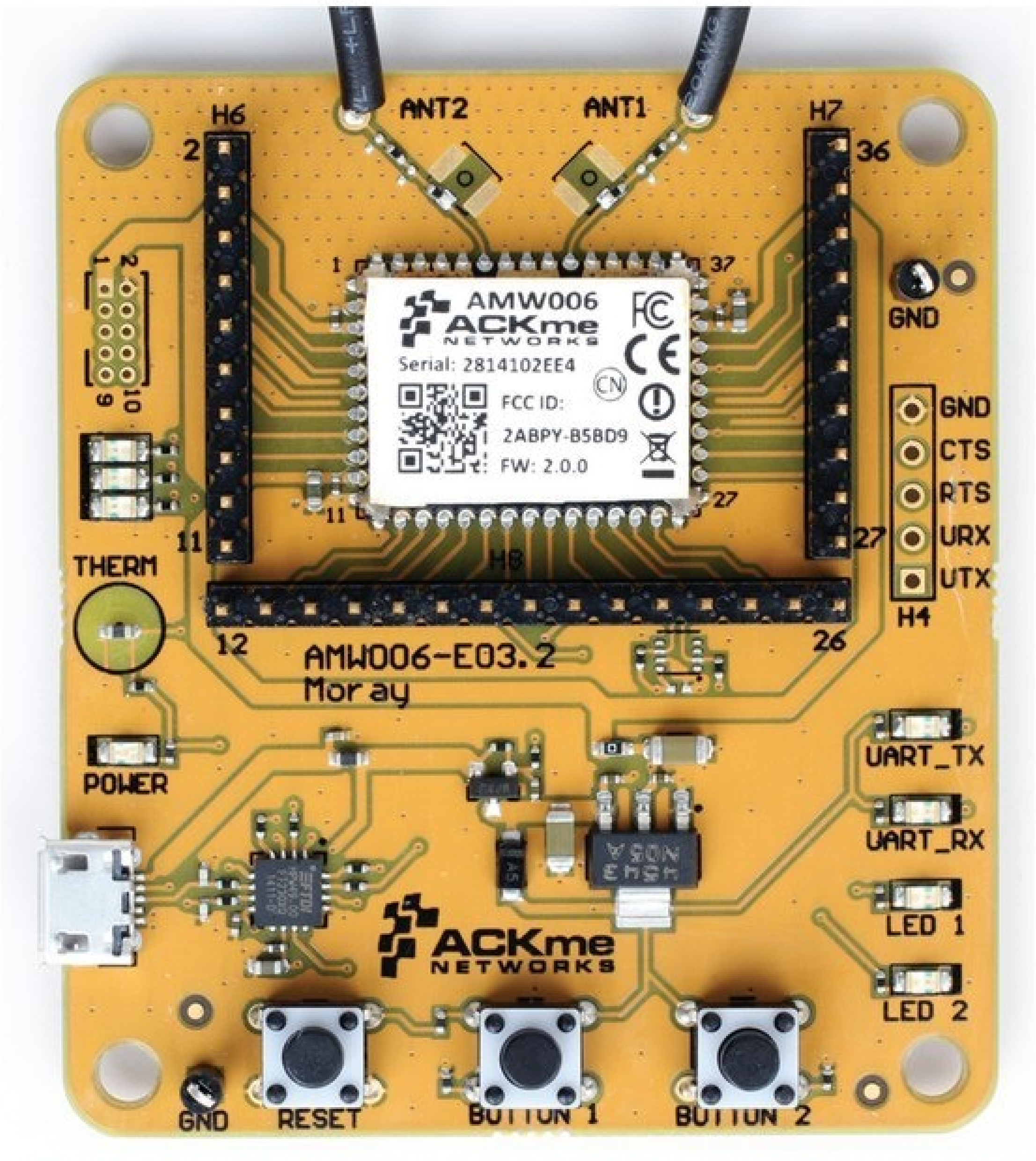
WiConnect provides a secure HTTP web server that is based on the RESTful API. In addition to the API, WiConnect includes an example web server application that is fully customizable. If you're an old head at web serving, you can apply your JavaScript and Python skills to WiConnect's convenient libraries.

I've been throwing the words HTTP and web server around like they're free. Well, they are free if you have a TCP/IP stack to back them up. The WiConnect system's TCP/IP stack has the ability to operate as a DHCP client or server, a DNS client or server, and a UDP/TCP/TLS client or server.

There is also support for the SNTP time protocol and an SMTP mailbox. If your web application requires security, you can employ WiConnect's SSL/TLS and HTTPS components. Wi-Fi is secured with the normal suspects: Open, WEP, WPA-PSK, and WPA2-PSK.

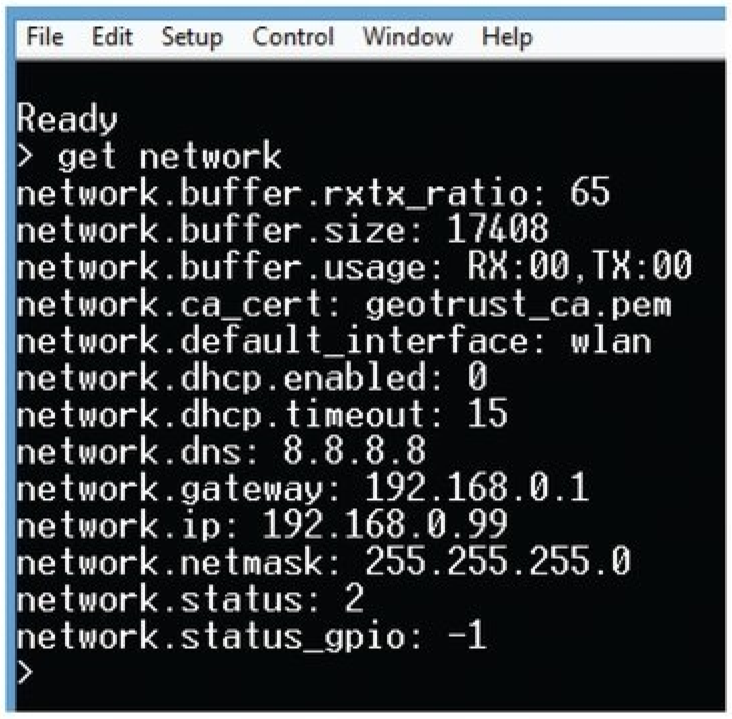
The Moray's Numbat Wi-Fi module is talented enough to utilize the power of its TCP/IP stack without the aid of an external microcontroller. The stack is supported by multiple serial ports and SPI portals. The API has the ability to mix the UDP, TCP, and HTTP communications sessions with the Numbat's native GPIO operations.

When it gets right down to it, that's why we're here.



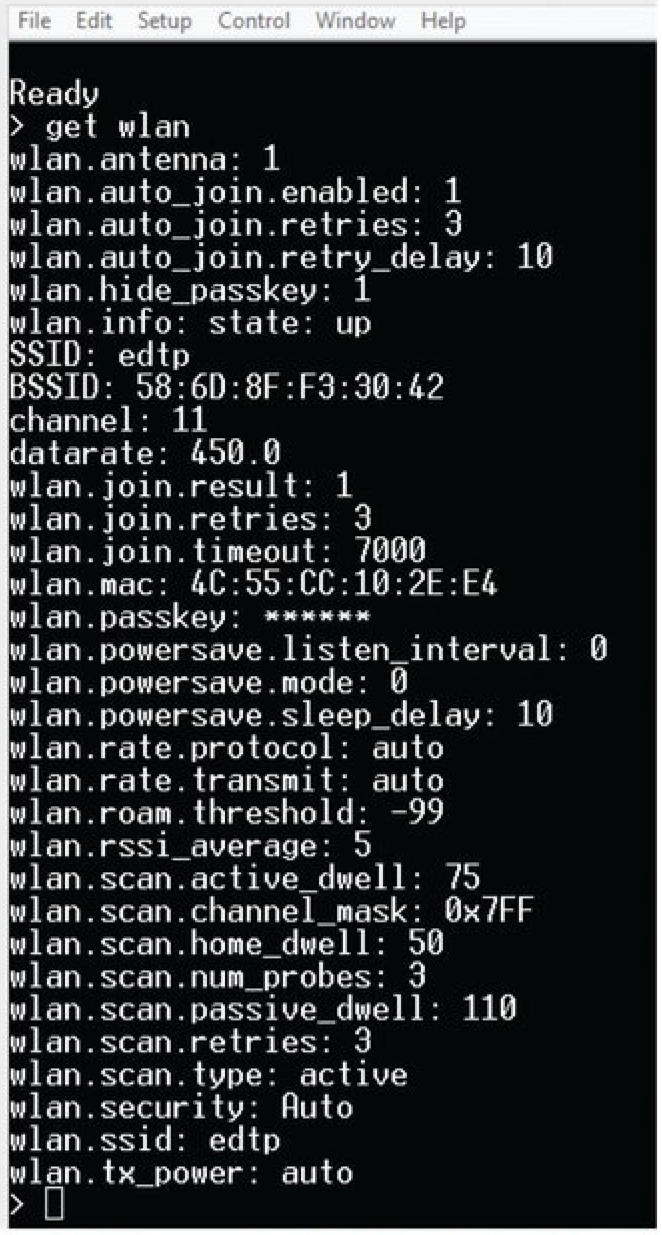
■ Photo 1. I discovered that I can hang a 1 MB serial Flash IC on the Numbat's master SPI portal. You can be sure I'll be trying that out in the very near future.

The idea is to use the Big Wire (Internet) as a pipe to channel data to and from a device using the well-worn Internet protocols. If we can do the same things to our device over the Internet as we do with the device wired in locally, the mission will be a success.



```
Ready
> get network
network.buffer.rxtx_ratio: 65
network.buffer.size: 17408
network.buffer.usage: RX:00,TX:00
network.ca_cert: geotrust_ca.pem
network.default_interface: wlan
network.dhcp.enabled: 0
network.dhcp.timeout: 15
network.dns: 8.8.8.8
network.gateway: 192.168.0.1
network.ip: 192.168.0.99
network.netmask: 255.255.255.0
network.status: 2
network.status_gpio: -1
>
```

■ Screenshot 2. This screen capture reveals the network configuration variables necessary to allow the Moray to join the EDTP LAN. Note that a static IP configuration is used instead of relying on DHCP to assign the network parameters.



```
Ready
> get wlan
wlan.antenna: 1
wlan.auto_join.enabled: 1
wlan.auto_join.retries: 3
wlan.auto_join.retry_delay: 10
wlan.hide_passkey: 1
wlan.info: state: up
SSID: edtp
BSSID: 58:6D:8F:F3:30:42
channel: 11
datarate: 450.0
wlan.join.result: 1
wlan.join.retries: 3
wlan.join.timeout: 7000
wlan.mac: 4C:55:CC:10:2E:E4
wlan.passkey: ******
wlan.powersave.listen_interval: 0
wlan.powersave.mode: 0
wlan.powersave.sleep_delay: 10
wlan.rate.protocol: auto
wlan.rate.transmit: auto
wlan.roam.threshold: -99
wlan.rssi_average: 5
wlan.scan.active_dwell: 75
wlan.scan.channel_mask: 0x7FF
wlan.scan.home_dwell: 50
wlan.scan.num_probes: 3
wlan.scan.passive_dwell: 110
wlan.scan.retries: 3
wlan.scan.type: active
wlan.security: Auto
wlan.ssid: edtp
wlan.tx_power: auto
> 
```

■ Screenshot 1. All of the WLAN configuration variables are displayed here. We need only to populate the variables that allow us to join the local EDTP LAN. Once on the LAN, we have access to the Big Wire.

The Setup

A server needs clients. Clients of the type we're looking for can be found lying naked on an electronics bench or embedded in a device attached to the Big Wire. The only way to access our clientele is to jump on that wire and listen. We need only speak when spoken to.

The Moray pictured in **Photo 1** can be accessed via its serial port, a Telnet connection, or via a web-based HTTP communications link. We will use the Moray's physical USB port to initially prepare the Moray's Numbat radio module for integration into an in-house LAN.

The Moray's serial port is based on an FTDI USB-to-serial IC. So, all we have to do is plug the Moray into almost any PC's host USB portal. If the PC is loaded with the FTDI drivers, a virtual COM port will be spawned. To use the newly born virtual COM port, we must make sure that the host PC has a suitable terminal emulator program installed. In our case, the host PC is running Windows 8.1, which is supporting the necessary FTDI drivers and a copy of the terminal emulator, Tera Term Pro.

The Moray's default baud rate is 115200 bps. The standard no parity, eight data bits, and one stop bit also apply. If everything is set up correctly in Tera Term Pro, after connecting the USB only a tap on the Enter key is required to invoke a *Ready* prompt from the Moray.

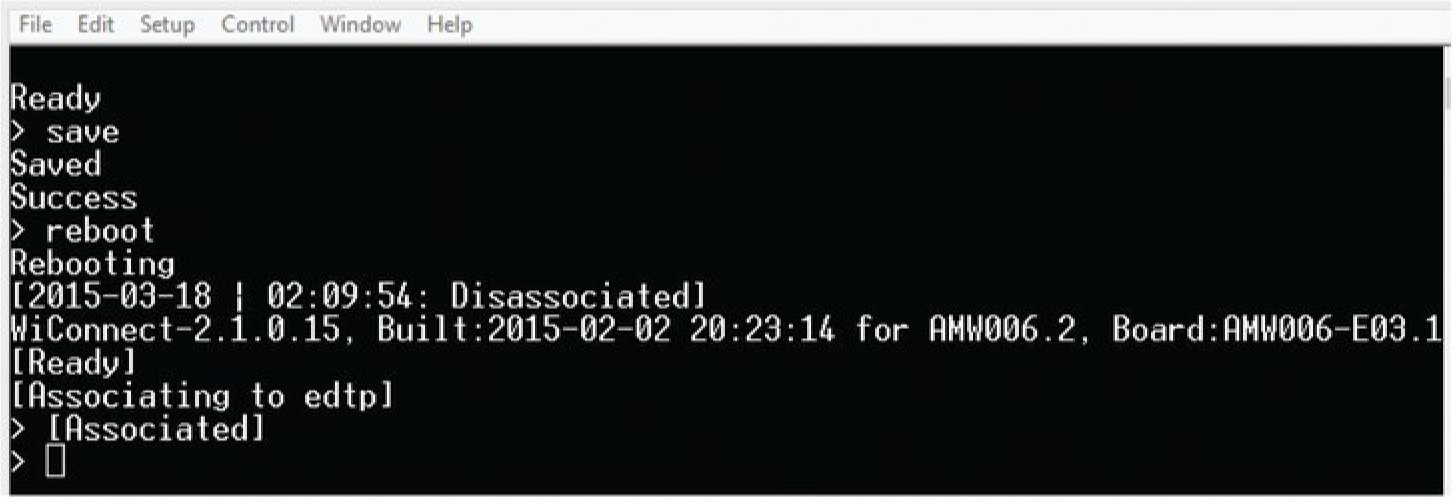
The contents of **Screenshot 1** contain the necessary configuration variables to get the Moray on the EDTP LAN. However, we haven't given the Moray enough network information to make the WLAN info stick. I've taken care of that in **Screenshot 2**.

Note that we are not using the router's DHCP server. Instead, we have configured the Moray to use a static IP address. The advantage to this is that we never have to guess about what IP address the Moray is operating against. Knowing the IP address also allows us to easily "open up" that IP address on the router for use as a server portal on the Big Wire.

In that I know everything about the EDTP LAN, all of our configuration work should immediately pay off. As you can see in **Screenshot 3**, I was right. A save and reboot operation resulted in an association of the Moray and the EDTP LAN. This is good as it paves the way for a web server to operate on the EDTP LAN.

At Your Service

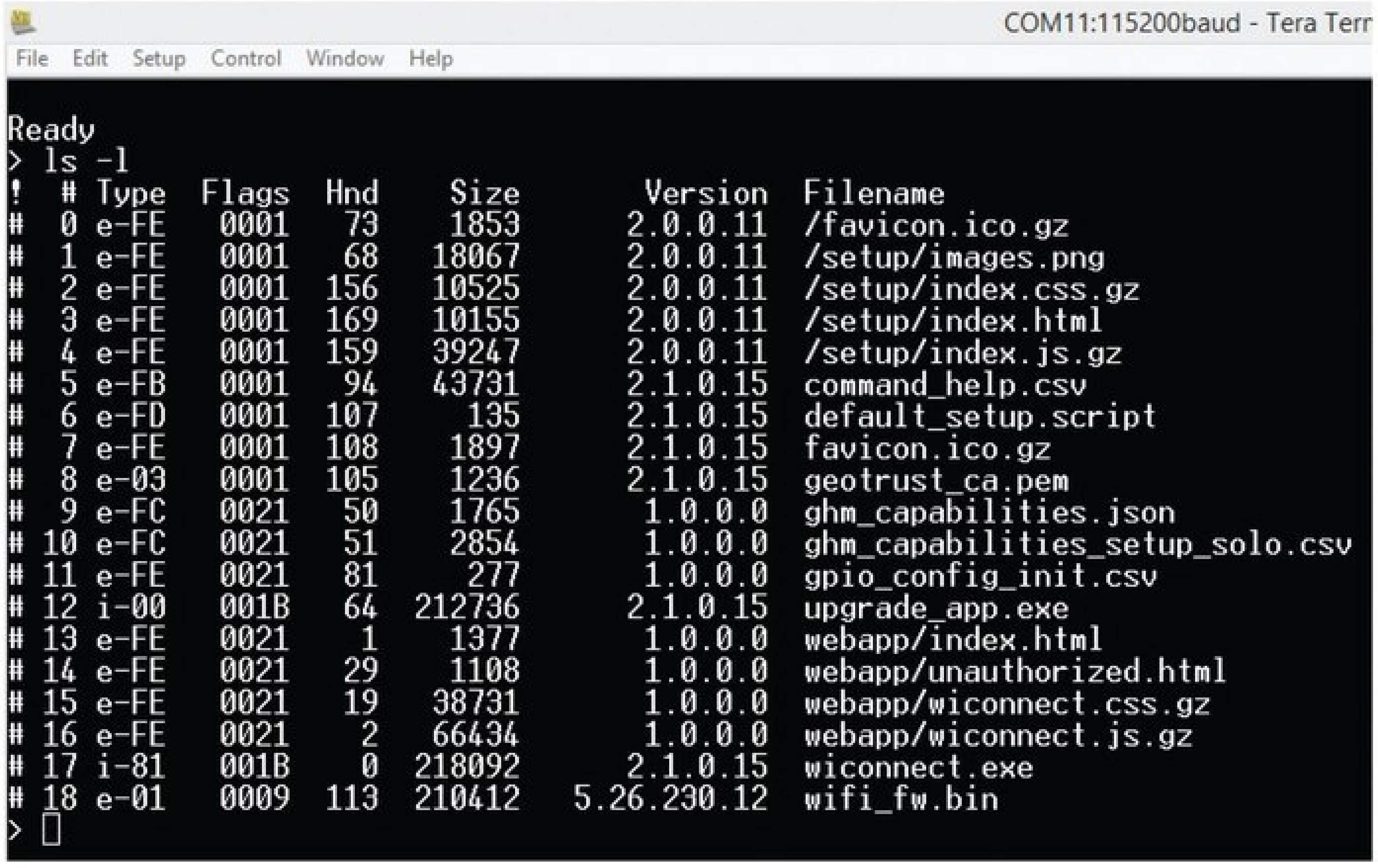
The folks at ACKme have developed a really neat web



```
Ready
> save
Saved
Success
> reboot
Rebooting
[2015-03-18 | 02:09:54: Disassociated]
WiConnect-2.1.0.15, Built:2015-02-02 20:23:14 for AMW006.2, Board:AMW006-E03.1
[Ready]
[Associating to edtp]
> [Associated]
> 
```

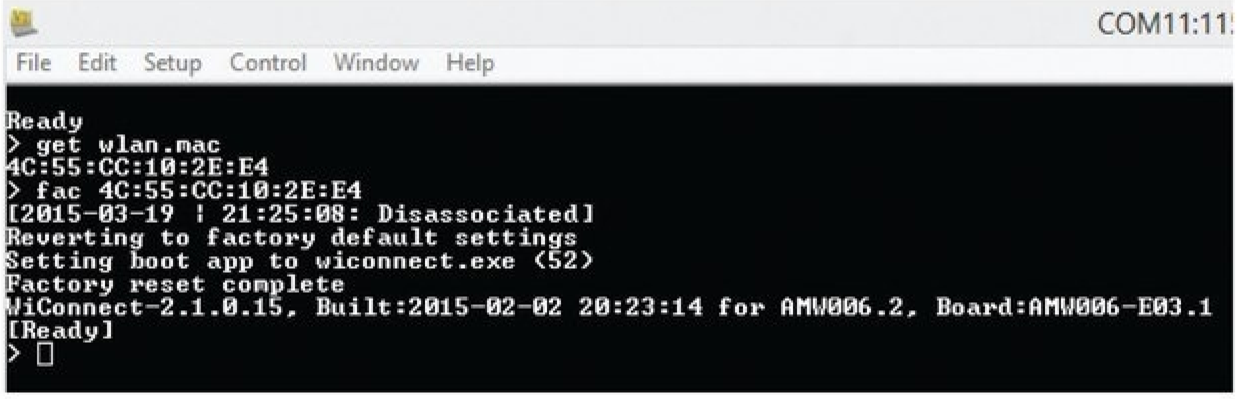
■ Screenshot 3. We are now on the "Little Wire," which is otherwise known as the EDTP LAN. The next step is to configure the Moray to operate as a web server on this wire.

Post comments on this article and find any associated files and/or downloads at **www.nutsvolts.com/index.php?/magazine/article/may2015_DesignCycle**.


COM11:115200baud - Tera Terr
File Edit Setup Control Window Help

```
Ready
> ls -l
!   # Type  Flags  Hnd    Size      Version  Filename
#   0 e-FE   0001   73    1853     2.0.0.11  /favicon.ico.gz
#   1 e-FE   0001   68   18067     2.0.0.11  /setup/images.png
#   2 e-FE   0001  156   10525     2.0.0.11  /setup/index.css.gz
#   3 e-FE   0001  169   10155     2.0.0.11  /setup/index.html
#   4 e-FE   0001  159   39247     2.0.0.11  /setup/index.js.gz
#   5 e-FB   0001   94   43731     2.1.0.15  command_help.csv
#   6 e-FD   0001  107     135     2.1.0.15  default_setup.script
#   7 e-FE   0001  108    1897     2.1.0.15  favicon.ico.gz
#   8 e-03   0001  105    1236     2.1.0.15  geotrust_ca.pem
#   9 e-FC   0021   50    1765      1.0.0.0  ghm_capabilities.json
#  10 e-FC   0021   51    2854      1.0.0.0  ghm_capabilities_setup_solo.csv
#  11 e-FE   0021   81     277      1.0.0.0  gpio_config_init.csv
#  12 i-00   001B   64  212736     2.1.0.15  upgrade_app.exe
#  13 e-FE   0021    1    1377      1.0.0.0  webapp/index.html
#  14 e-FE   0021   29    1108      1.0.0.0  webapp/unauthorized.html
#  15 e-FE   0021   19   38731      1.0.0.0  webapp/wiconnect.css.gz
#  16 e-FE   0021    2   66434      1.0.0.0  webapp/wiconnect.js.gz
#  17 i-81   001B    0  218092     2.1.0.15  wiconnect.exe
#  18 e-01   0009  113  210412   5.26.230.12  wifi_fw.bin
> □
```

■ Screenshot 4. I smell Linux here. The listing represents the contents of the Numbat's internal file system.



```
Ready
> get wlan.mac
4C:55:CC:10:2E:E4
> fac 4C:55:CC:10:2E:E4
[2015-03-19 | 21:25:08: Disassociated]
Reverting to factory default settings
Setting boot app to wiconnect.exe (52)
Factory reset complete
WiConnect-2.1.0.15, Built:2015-02-02 20:23:14 for AMW006.2, Board:AMW006-E03.1
[Ready]
> □
```

■ Screenshot 5. To perform a factory reset, we must know the module's MAC address. As you can see, we obtain the MAC address with the *get wlan.mac* command. You don't see it here, but just for grins I performed a reboot following the factory reset operation.

server application that we will take a look at in a moment. Right now, let's see just how easy it is to set up and serve a simple web page using the Moray.

Behold **Screenshot 4**. I have displayed the contents of the Numbat's internal file system by issuing the WiConnect *ls -l* command. Some of the files have a familiar feel. For instance, files with the extension of .html are HTML page files that are used by the web server. The files called *index.html* are the "front pages" of a particular web application.

The *setup* web application is fronted by */setup/index.html*. It looks like the */setup* directory holds its own *index.html* file, and the */webapp* directory does likewise with its particular *index.html* file entry.

The Numbat's WiConnect file system does not really support directories. However, when it comes to designating directories and files, just do as you have always done and the WiConnect's file system management components will take care of the rest.

The Numbat's memory is divided into three distinct parts. The internal Flash is found within the Numbat's microprocessor. An extended Flash IC exists under the module's metal shield that is under the control of the user via the API. In an embedded application, memory is always a coveted resource.

To augment the Numbat's standard memory configuration, the Numbat can be fitted with additional external bulk Flash.

The Numbat specs tell us that up to 128 MB of bulk Flash can be supported. Bulk Flash — like extended Flash — is serial Flash. Several types of serial Flash ICs are currently supported.

Macronix has a 1 MB (MX25L8006E) and 2 MB (MX25L1606E) Flash memory device that is certified. Microchip has a 1 MB Flash device (SST25Vf080B) on the list, and Eon Silicon Solution adds two more 1 MB Flash ICs to the approved list (EN25QH16 and en25Q80B). All of the aforementioned serial Flash ICs interface to the Numbat Wi-Fi module via its SPI master portal.

All we have to do to activate the new bulk Flash is specify which Numbat GPIO we want to act as the serial Flash chip select pin. We perform the chip select pin selection by executing the *system.bflash.cs_gpio* command and populating its associated configuration variable with the desired GPIO pin number using a set command. For instance, let's assign GPIO_1 to the chip select duty: *set system.bflash.cs_gpio 1*.

The default Moray hardware configuration contains a 1 MB chunk of serial Flash. That's more than enough for us to assemble, load, and display a nice and pretty *index.html* page.

From the Beginning

Let's start with a clean slate. The first thing we'll do is force a factory reset. The factory reset sequence is shown in **Screenshot 5**. We will have to use the *set* command sequences to manually reload our *wlan* and *network*

RESOURCES

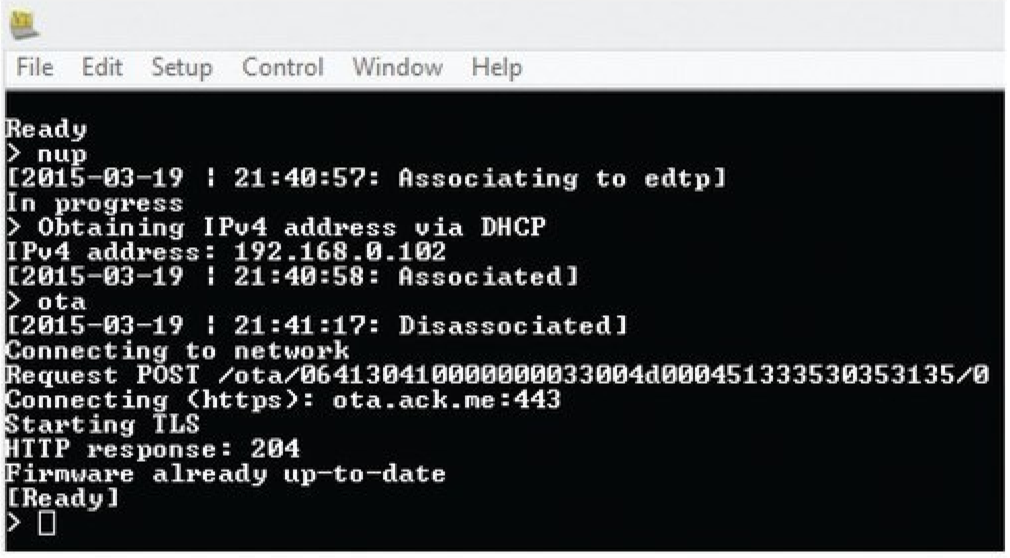
ACKme
Moray Development Board
Numbat Wi-Fi Module
www.ack.me

Lemos International
www.lemosint.com

configuration variables. There are “automatic” ways to bring our Moray up, but I prefer the old school “I know how because I know why” methodology. I entered just enough network information (SSID and passkey) to allow the Moray to pull itself up using DHCP.

As you can see in **Screenshot 6**, I urged the network up with the *nup* (network_up) command. To be sure that everything was up-to-date, I then issued the *ota* command, which reaches out to the ACKme server and downloads the latest Numbat firmware. At this point, I disabled DHCP (*set network.dhcp.enabled 0*) and set up the static gateway, IP, and netmask values using these *set static* commands:

```
set static.gateway 192.168.0.1
set static.ip 192.168.0.99
set static.netmask 255.255.255
```

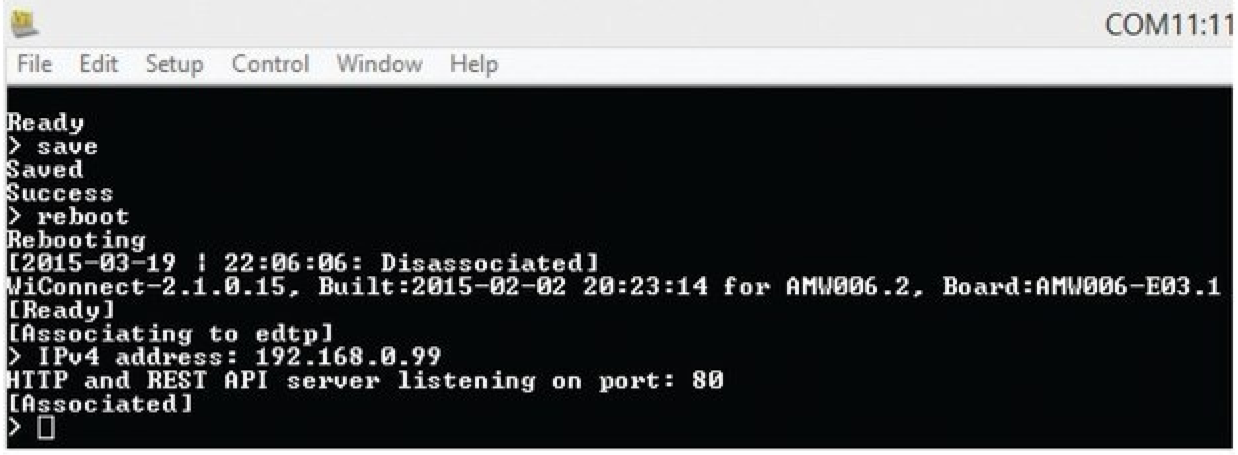


```
Ready
> nup
[2015-03-19 | 21:40:57: Associating to edtp]
In progress
> Obtaining IPv4 address via DHCP
IPv4 address: 192.168.0.102
[2015-03-19 | 21:40:58: Associated]
> ota
[2015-03-19 | 21:41:17: Disassociated]
Connecting to network
Request POST /ota/06413041000000000033004d0004513335303531 35/0
Connecting (https): ota.ack.me:443
Starting TLS
HTTP response: 204
Firmware already up-to-date
[Ready]
> 
```

■ Screenshot 6. Once the Moray is on a network, we can issue the OTA (over the air) command to update its firmware by way of the remote master ACKme server.

While I was on a roll, I used the *set* commands to enable auto join, hide the passkey (for enhanced protection against hackers), and enable the server function:

```
set wlan.auto_join.enabled 1
set wlan.hide_passkey 1
set http.server.enabled 1
```



```
Ready
> save
Saved
Success
> reboot
Rebooting
[2015-03-19 | 22:06:06: Disassociated]
WiConnect-2.1.0.15, Built:2015-02-02 20:23:14 for AMW006.2, Board:AMW006-E03.1
[Ready]
[Associating to edtp]
> IPv4 address: 192.168.0.99
HTTP and REST API server listening on port: 80
[Associated]
> 
```

■ Screenshot 7. The factory-fresh Moray development board is now officially up on the EDTP LAN operating as a web server, listening on port 80.

We should be able to call up the *index.html* file that represents the default web application by kicking off a browser and entering *http://192.168.0.99* in the browser's URL box. All went well, and **Screenshot 8** appeared.

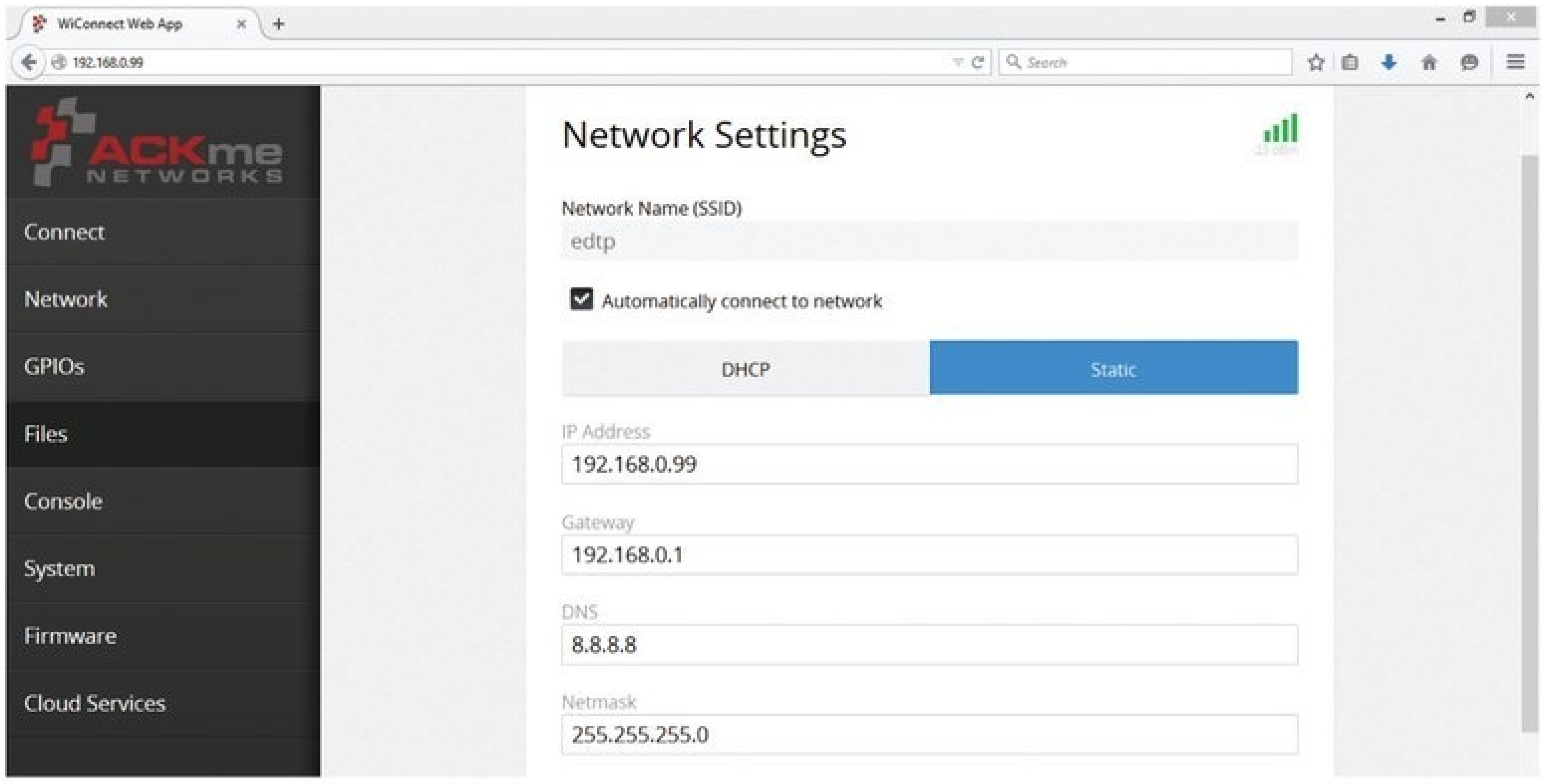
Now that we have a working web server running on our Moray, we can turn our attention to building a simple web page that will ultimately be loaded into the Numbat's external Flash memory.

I'm a Dreamweaver fan. So, to make it super simple, I'm going to drop a smaller version of **Photo 1** into a new HTML document, add some text, and save it as *indexM.html*. A quick look at my make-shift *indexM.html* file from a Dreamweaver point-of-view is shown in **Screenshot 9**.

The example web application demonstrates the capabilities of the WiConnect API. So, we're going to take advantage of the example web application and use its file transfer functionality to place our newly minted *indexM.html* file into the Numbat's external serial Flash.

As you can see in **Screenshot 10**, it only takes a point and a click to load the file. We'll need to have the image of the Moray available for the *indexM.html* available in the Numbat's Flash, as well. I'll load it in the same manner.

Okay. The desired files are loaded. However, they will not be called until we tell WiConnect to use them. As it



Network Settings

Network Name (SSID)
edtp

☑ Automatically connect to network

DHCP | Static

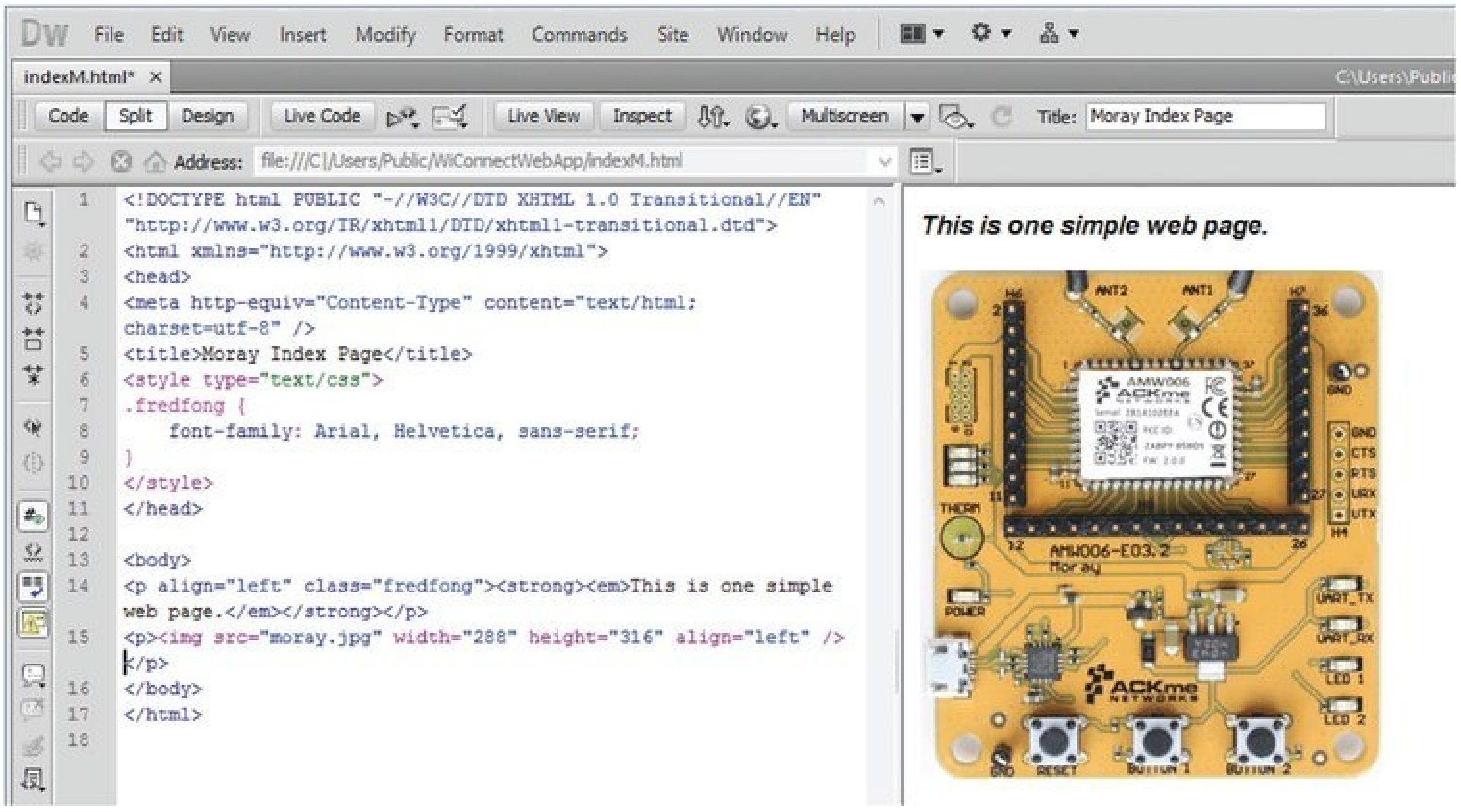
IP Address
192.168.0.99

Gateway
192.168.0.1

DNS
8.8.8.8

Netmask
255.255.255.0

■ Screenshot 8. This is the *index.html* page that represents the example web application that is shipped with the Numbat's operational firmware.


Dw File Edit View Insert Modify Format Commands Site Window Help
indexM.html*
Code Split Design Live Code Live View Inspect Multiscreen
Title: Moray Index Page
Address: file:///C|/Users/Public/WiConnectWebApp/indexM.html
This is one simple web page.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=utf-8" />
<title>Moray Index Page</title>
<style type="text/css">
.fredfong {
    font-family: Arial, Helvetica, sans-serif;
}
</style>
</head>

<body>
<p align="left" class="fredfong"><strong><em>This is one simple
web page.</em></strong></p>
<p><img src="moray.jpg" width="288" height="316" align="left" />
</p>
</body>
</html>
```

■ Screenshot 9. Using Dreamweaver is as easy or as complicated as you want it to be. This little web page took about two minutes to construct.

stands, every time the Moray is booted, the *index.html* file in the *webapp* directory will be used. We can force our new *indexM.html* to be called by specifying it with a *set* command in this manner:

```
set http.server.root_filename indexM.html
```

Once the root file name has been reassigned, every time **http://192.168.0.99** is addressed, our new *indexM.html* page will be displayed first.

If you don't believe me, the image of the web page captured in **Screenshot 11** looks a lot like our Dreamweaver layout.

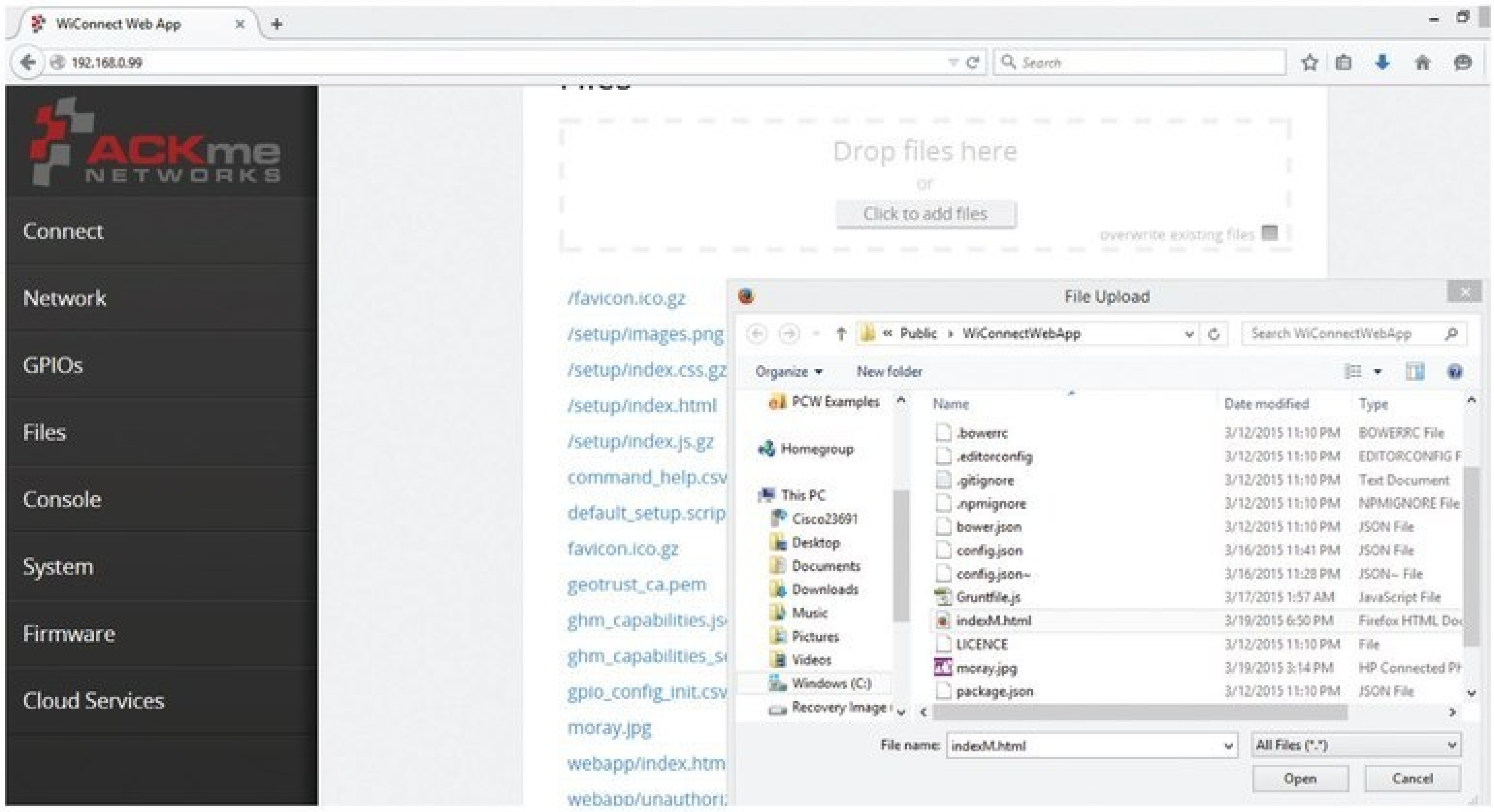
It Gets Better from Here

It's really easy to get the Moray up on a LAN, and just as easy to turn it into a web server. A few clicks of the mouse and you're there. You can dig deeper into the WiConnect world by visiting **wiconnect.ack.me**. Once there, you will want to read and absorb as much as you can.

When you think your sponge is saturated, find the topic *Customizing the WiConnect Web App*. You will be introduced to the open source web tools used to build the example web application.

If you decide to go further down the WiConnect road, you'll find yourself downloading, installing, and running some really neat open source web development tools. I think you will really enjoy learning about and using Grunt. Grunt is like a DOS batch file on steroids.

I recommend going to the Grunt website (**www.gruntjs.com**) and getting familiar with how Grunt works. Then, look at the Grunt file that comes with the web app code you can download from Git Hub. You'll soon see how WiConnect API functions are defined and called from Grunt. As you delve deeper into the web server tools, you'll come to find that the JavaScript server code included in the download package that is called from Grunt can be very useful beyond its utilization in the customizing application.

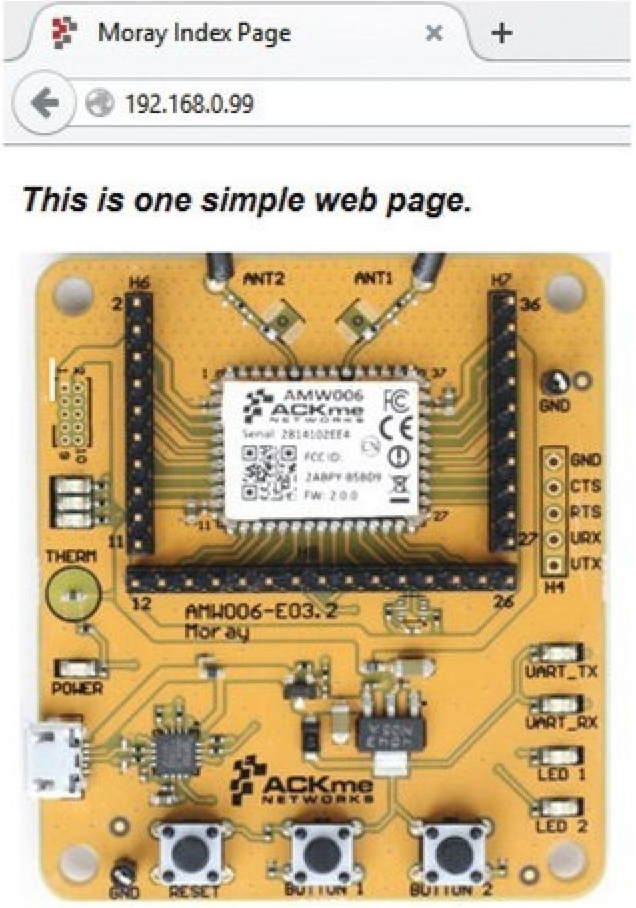


/favicon.ico.gz
/setup/images.png
/setup/index.css.gz
/setup/index.html
/setup/index.js.gz
command_help.csv
default_setup.scrip
favicon.ico.gz
geotrust_ca.pem
ghm_capabilities.js
ghm_capabilities_s
gpio_config_init.csv
moray.jpg
webapp/index.htm
webapp/unauthori

Name	Date modified	Type
.bowerrc	3/12/2015 11:10 PM	BOWERRC File
.editorconfig	3/12/2015 11:10 PM	EDITORCONFIG F
.gitignore	3/12/2015 11:10 PM	Text Document
.npmignore	3/12/2015 11:10 PM	NPMIGNORE File
bower.json	3/12/2015 11:10 PM	JSON File
config.json	3/16/2015 11:41 PM	JSON File
config.json~	3/16/2015 11:28 PM	JSON~ File
Gruntfile.js	3/17/2015 1:57 AM	JavaScript File
indexM.html	3/19/2015 6:50 PM	Firefox HTML Do
LICENCE	3/12/2015 11:10 PM	File
moray.jpg	3/19/2015 3:14 PM	HP Connected Ph
package.json	3/12/2015 11:10 PM	JSON File

File name: indexM.html
All Files (*.*)

■ Screenshot 10. The example web application calls the RESTful API's *http_download* function to perform the transfer of the files from the PC to the Numbat's Flash.



This is one simple web page.

■ Screenshot 11. Service with a smile. The Moray web server was as easy to implement as the web page itself.

When it comes to learning about and using the Moray, the most expensive investment is in your PC. The web development software is free for the most part. All you need to purchase to get started is a Moray development board.

Once you have a Moray attached to your PC, you can issue WiConnect commands and experiment with embedded web serving until the cows come home. By the way, you can also add wireless ACKme embedded web serving to your Design Cycle. **NV**

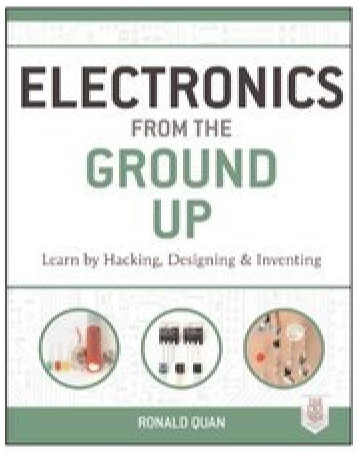
For complete product details, visit our webstore!!

The Nuts & Volts WEBSTORE

GREAT FOR DIYers!

Electronics from the Ground Up: Learn by Hacking, Designing, and Inventing

by Ronald Quan


ELECTRONICS FROM THE GROUND UP
Learn by Hacking, Designing & Inventing
RONALD QUAN

Are you fascinated by the power of even the smallest electronic device? Electronics from the Ground Up guides you through step-by-step experiments that reveal how electronic circuits function so you can advance your skills and design custom circuits. You'll work with a range of circuits and signals related to optical emitters and receivers, audio, oscillators, and video.

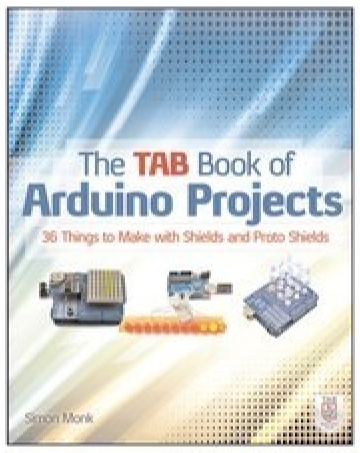
Paper back 544 pages.

$30.00

The TAB Book of Arduino Projects

by Simon Monk

The ultimate collection of DIY Arduino projects!


The TAB Book of Arduino Projects
36 Things to Make with Shields and Proto Shields
Simon Monk

In this easy-to-follow book, electronics guru Simon Monk shows you how to create a wide variety of fun and functional gadgets with the Arduino Uno and Leonardo boards. Filled with step-by-step instructions and detailed illustrations, The TAB Book of Arduino Projects: 36 Things to Make with Shields and Proto Shields provides a cost estimate, difficulty level, and list of required components for each project.

$30.00

Arduino Projects for Amateur Radio

by Jack Purdum, Dennis Kidder

Boost Your Ham Radio's Capabilities Using Low Cost Arduino Microcontroller Boards

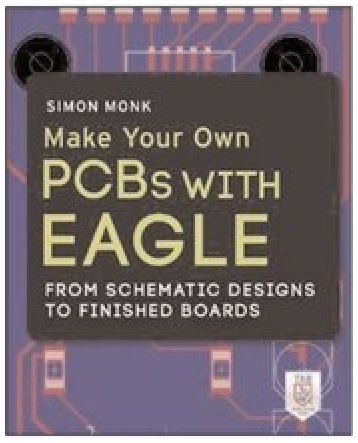
ARDUINO PROJECTS for AMATEUR RADIO
Dr. Jack Purdum and Dennis Kidder

Do you want to increase the functionality and value of your ham radio without spending a lot of money? This book will show you how! *Arduino Projects for Amateur Radio* is filled with step-by-step microcontroller projects you can accomplish on your own — no programming experience necessary.

$30.00

Make Your Own PCBs with EAGLE

by Eric Kleinert


SIMON MONK
Make Your Own PCBs WITH EAGLE
FROM SCHEMATIC DESIGNS TO FINISHED BOARDS

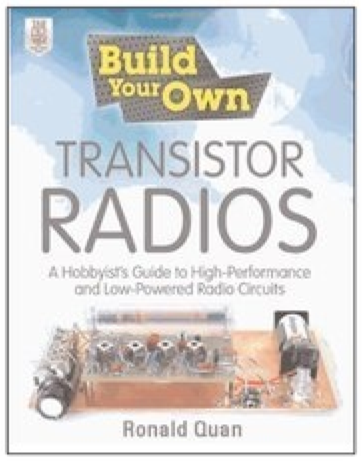
Featuring detailed illustrations and step-by-step instructions, *Make Your Own PCBs with EAGLE* leads you through the process of designing a schematic and transforming it into a PCB layout. You'll then move on to fabrication via the generation of standard Gerber files for submission to a PCB manufacturing service. This practical guide offers an accessible, logical way to learn EAGLE and start producing PCBs as quickly as possible.

$30.00

Build Your Own Transistor Radios

by Ronald Quan

A Hobbyist's Guide to High Performance and Low-Powered Radio Circuits


Build Your Own TRANSISTOR RADIOS
A Hobbyist's Guide to High-Performance and Low-Powered Radio Circuits
Ronald Quan

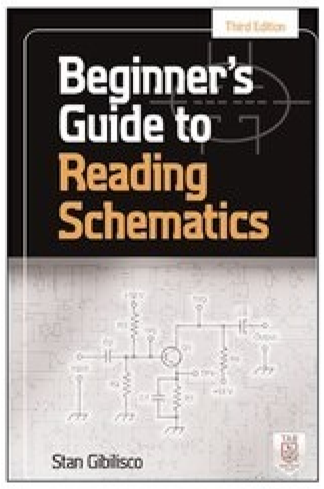
Create sophisticated transistor radios that are inexpensive yet highly efficient. Inside this book, it offers complete projects with detailed schematics and insights on how the radios were designed. Learn how to choose components, construct the different types of radios, and troubleshoot your work.

Paperback, 496 pages

$49.95

Beginner's Guide to Reading Schematics, 3E

by Stan Gibilisco



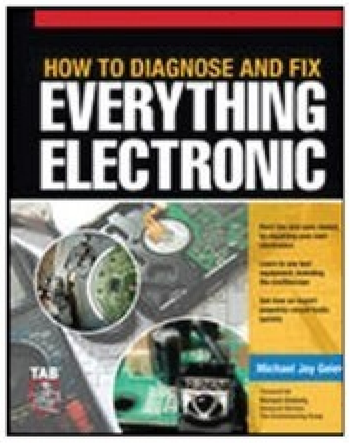
Navigate the roadmaps of simple electronic circuits and complex systems with help from an experienced engineer. With all-new art and demo circuits you can build, this hands-on, illustrated guide explains how to understand and create high-precision electronics diagrams. Find out how to identify parts and connections, decipher element ratings, and apply diagram-based information in your own projects.

$25.00

How to Diagnose and Fix Everything Electronic

by Michael Jay Geier

Master the Art of Electronics Repair

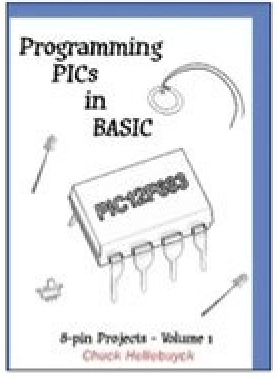


In this hands-on guide, a lifelong electronics repair guru shares his tested techniques and invaluable insights. *How to Diagnose and Fix Everything Electronic* shows you how to repair and extend the life of all kinds of solid-state devices, from modern digital gadgetry to cherished analog products of yesteryear.

$24.95

Programming PICs in Basic

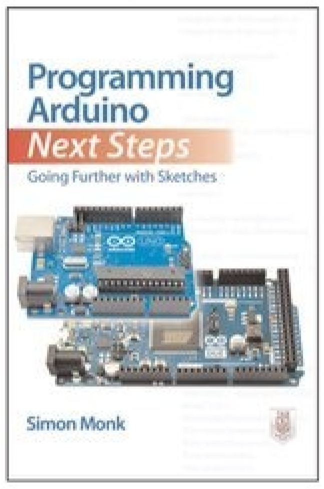
by Chuck Hellebuyck



If you wanted to learn how to program microcontrollers, then you've found the right book! Microchip PIC microcontrollers are being designed into electronics throughout the world and none is more popular than the eight-pin version. Now the home hobbyist can create projects with these little microcontrollers using a low cost development tool called the CHIPAXE system and the Basic software language. Chuck Hellebuyck introduces how to use this development setup to build useful projects with an eight-pin PIC12F683 microcontroller. **$14.95**

Programming Arduino Next Steps: Going Further with Sketches

by Simon Monk



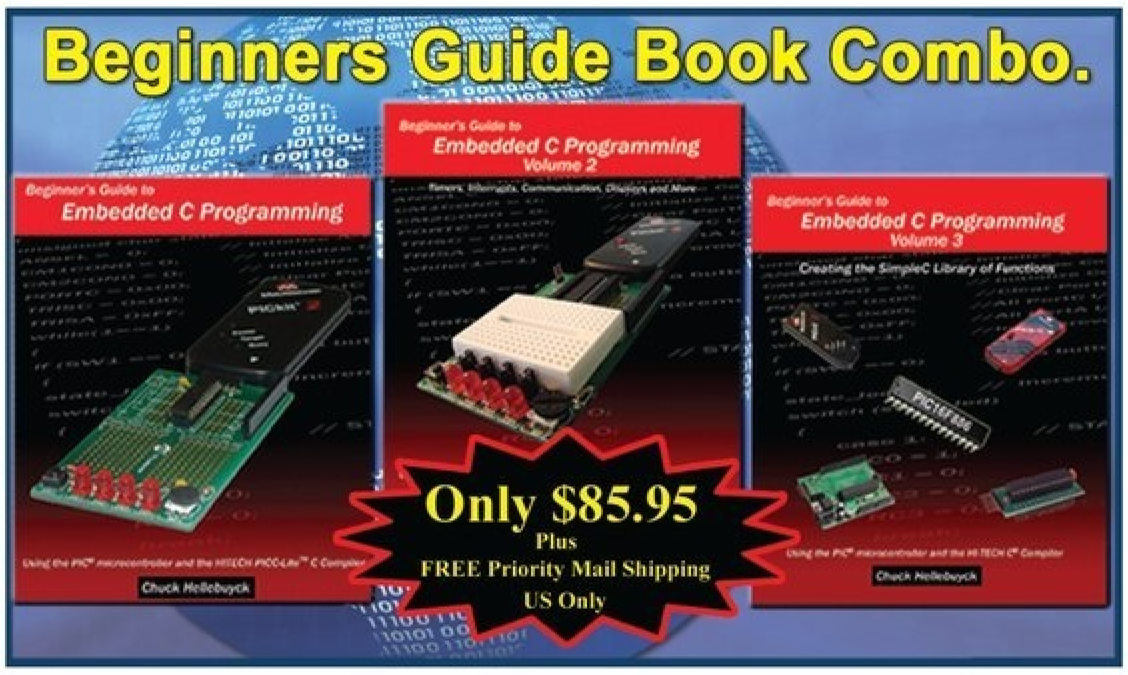
In this practical guide, electronics guru Simon Monk takes you under the hood of Arduino and reveals professional programming secrets. Also shows you how to use interrupts, manage memory, program for the Internet, maximize serial communications, perform digital signal processing, and much more. All of the 75+ example sketches featured in the book are available for download. **$20.00**

Order online @ www.store.nutsvolts.com

Or CALL 1-800-783-4624 today!

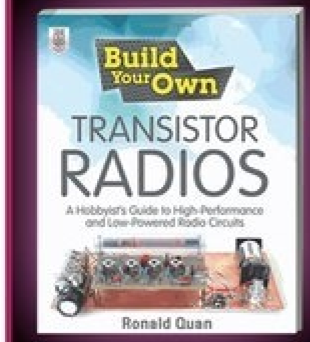
EDUCATIONAL

Beginners Guide Book Combo.

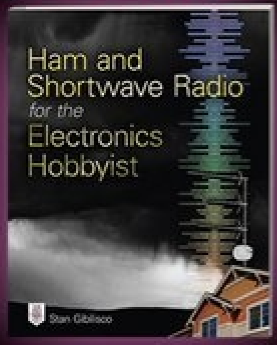

Beginner's Guide to
Embedded C Programming
Using the PIC® microcontroller and the HI-TECH PICC-Lite™ C Compiler
Chuck Hellebuyck
Beginner's Guide to
Embedded C Programming
Volume 2
Timers, Interrupts, Communication, Displays and More
Beginner's Guide to
Embedded C Programming
Volume 3
Creating the SimpleC Library of Functions
PIC16F886
Using the PIC® microcontroller and the HI-TECH C® Compiler
Chuck Hellebuyck

Only $85.95
Plus
FREE Priority Mail Shipping
US Only

DO YOU LOVE RADIOS?


Build Your Own
TRANSISTOR RADIOS
A Hobbyist's Guide to High-Performance and Low-Powered Radio Circuits
Ronald Quan

Sale Price
$39.95



Sale Price
$19.95

ARDUINO PROJECTS for AMATEUR RADIO
Dr. Jack Purdum and Dennis Kidder

Sale Price
$23.95

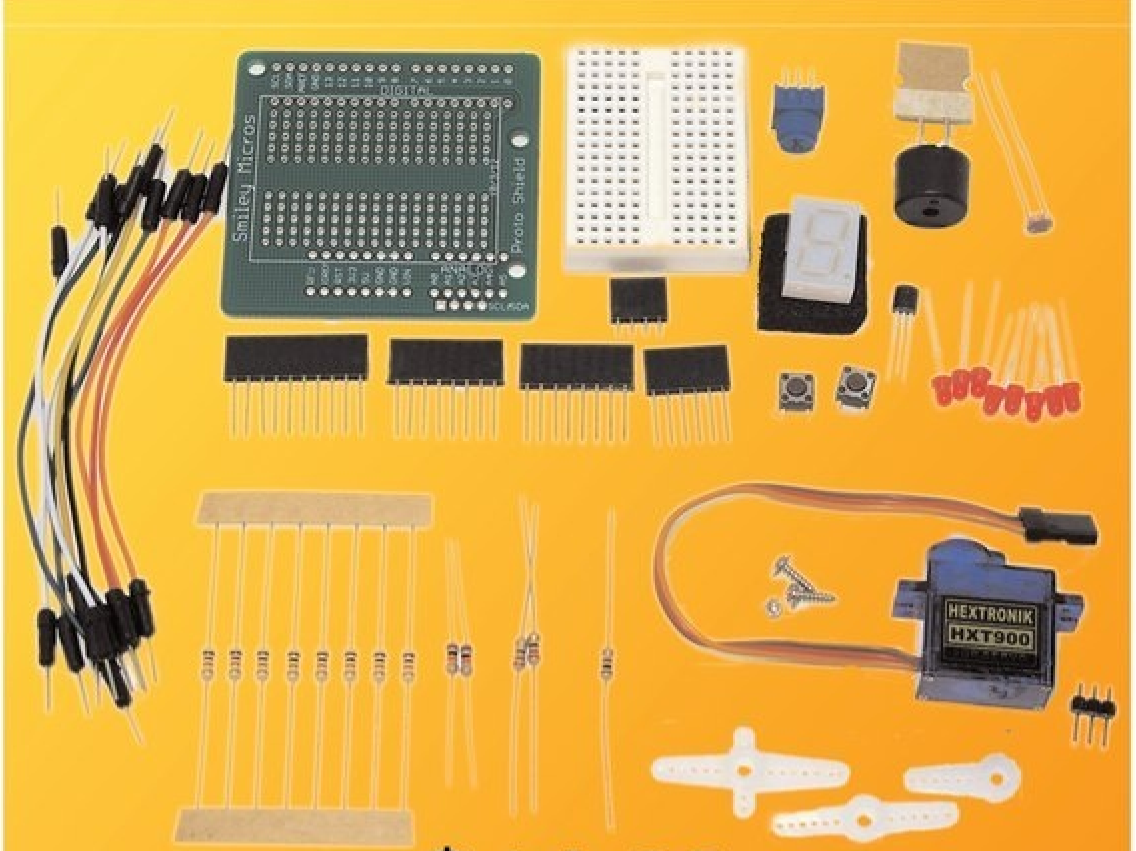
Get 20% off these three books PLUS Free Std. Shipping!

To order call 800 783-4624 or visit: http://store.nutsvolts.com

Limited Time Only

Arduino Classroom

Arduino 101 Projects Kit



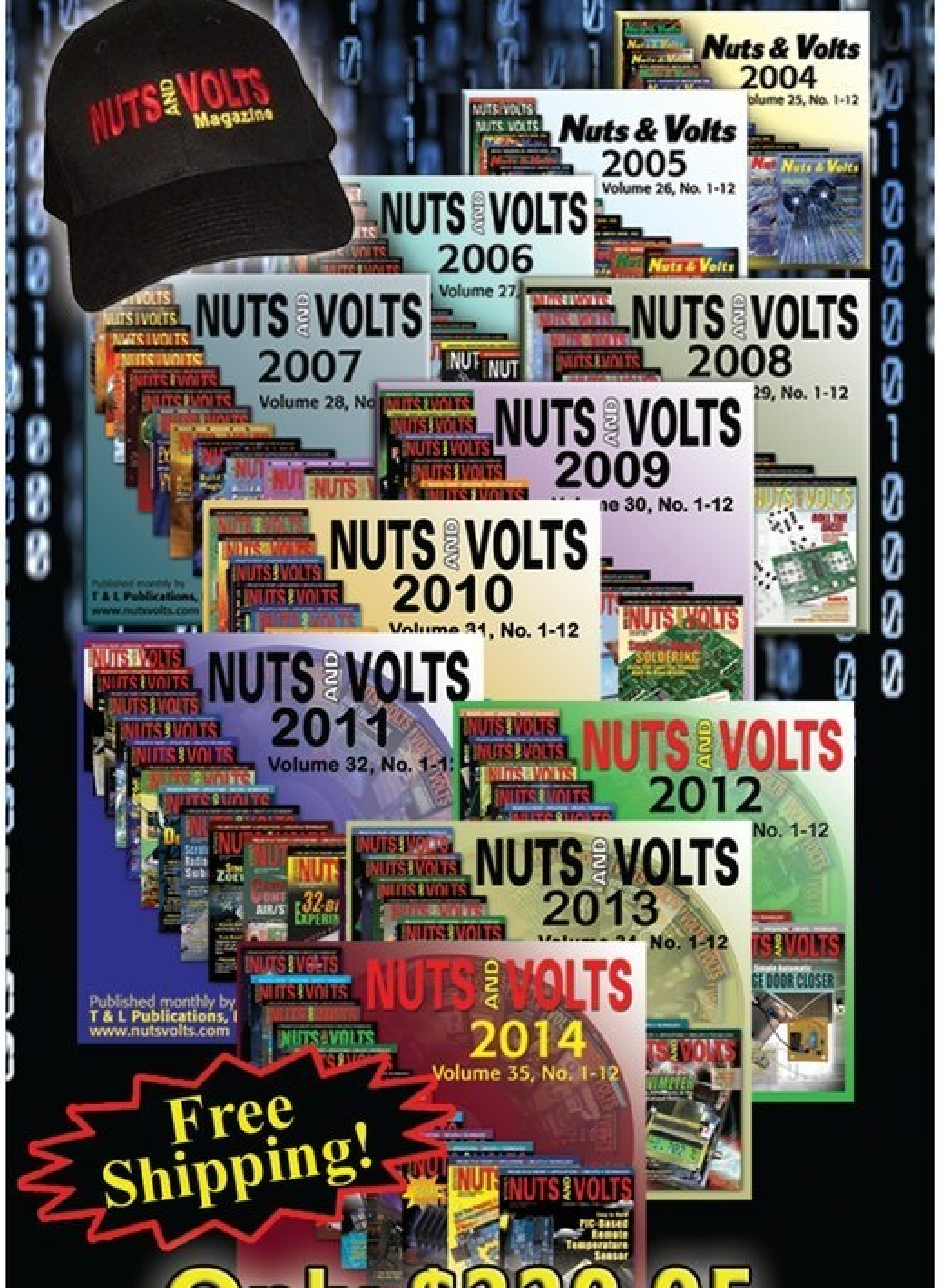
$44.99

From Smiley's Workshop

CD-ROM SPECIAL

Nuts & Volts 11 CD-ROMs & Hat Special!

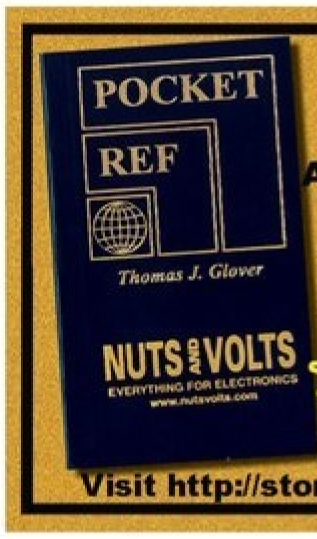
That's 132 issues.
Complete with supporting code and media files.


NUTS AND VOLTS Magazine
Nuts & Volts 2004 Volume 25, No. 1-12
Nuts & Volts 2005 Volume 26, No. 1-12
NUTS AND VOLTS 2006 Volume 27
NUTS AND VOLTS 2007 Volume 28, No
NUTS AND VOLTS 2008 29, No. 1-12
NUTS AND VOLTS 2009 ne 30, No. 1-12
NUTS AND VOLTS 2010 Volume 31, No. 1-12
NUTS AND VOLTS 2011 Volume 32, No. 1-1
NUTS AND VOLTS 2012 No. 1-12
NUTS AND VOLTS 2013 No. 1-12
NUTS AND VOLTS 2014 Volume 35, No. 1-12
Published monthly by T & L Publications, Inc. www.nutsvolts.com

Free Shipping!

Only $229.95
or $24.95 each.

The Nuts & Volts Pocket Ref



All the info you need at your fingertips!

This great little book is a concise all-purpose reference featuring hundreds of tables, maps, formulas, constants & conversions.

Only $12.95

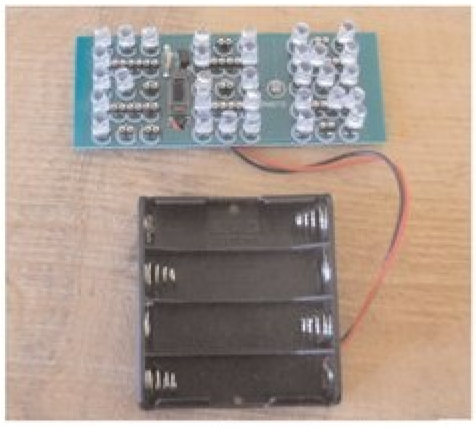
AND it still fits in your shirt pocket!

Visit http://store.nutsvolts.com or call (800) 783-4624

CALL 1-800-783-4624 today!
Order online @ www.nutsvolts.com

PROJECTS

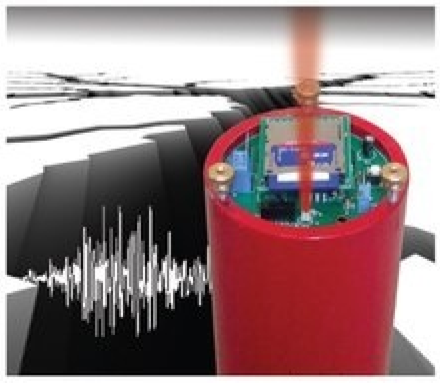
No Nonsense Annunciator Kit



The no nonsense/no microprocessor annunciator is a great little circuit that helps you get your message out without spending too much money. Put two circuits together and you'll have a six letter annunciator!
This kit is also a fun project to refine your soldering skills with its 102 socket pin connection points. WOW, that's a lot of soldering!

$19.95

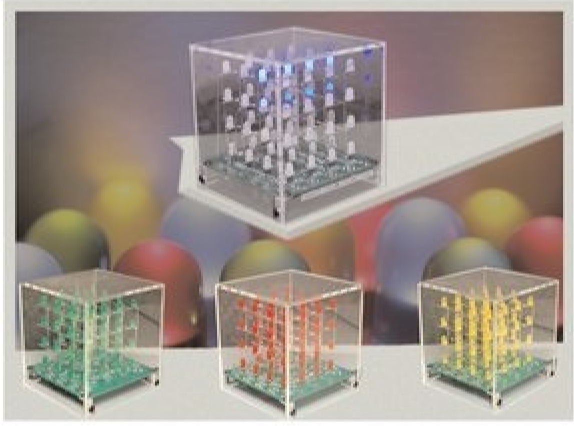
Seismograph Kit



Now you can record your own shaking, rattling, and rolling.
The Poor Man's Seismograph is a great project/device to record any movement in an area where you normally shouldn't have any. The kit includes everything needed to build the seismograph. All you need is your PC, SD card, and to download the free software to view the seismic event graph.

$79.95

3D LED Cube Kit



This kit shows you how to build a really cool 3D cube with a 4 x 4 x 4 monochromatic LED matrix which has a total of 64 LEDs. The preprogrammed microcontroller that includes 29 patterns that will automatically play with a runtime of approximately 6-1/2 minutes.
Colors available: Green, Red, Yellow & Blue

$57.95

Solar Charge Controller Kit 2.0

If you charge batteries using solar panels, then you can't afford not to have them protected from over-charging. This 12 volt/12 amp charge controller is great protection for the money. It is simple to build, ideal for the novice, and no special tools are needed other than a soldering iron and a 9/64" drill!

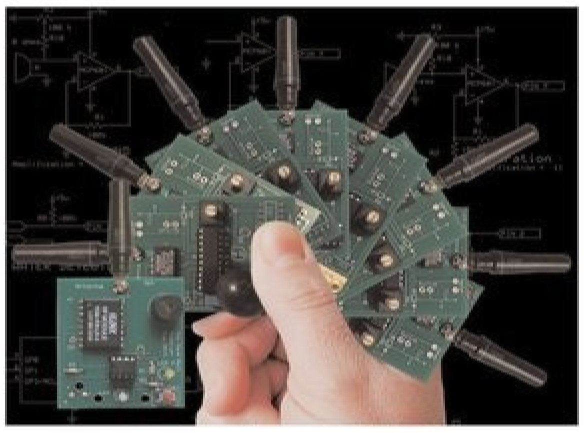
$27.95

Geiger Counter Kit

This kit is a great project for high school and university students The unit detects and displays levels of radiation, and can detect and display dosage levels as low as one micro-roentgen/hr.
The LND 712 tube in our kit is capable of measuring alpha, beta, and gamma particles.
Partial kits also available.

$159.95

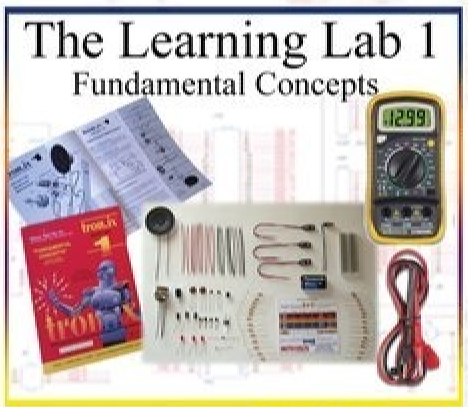
Super Detector Circuit Set



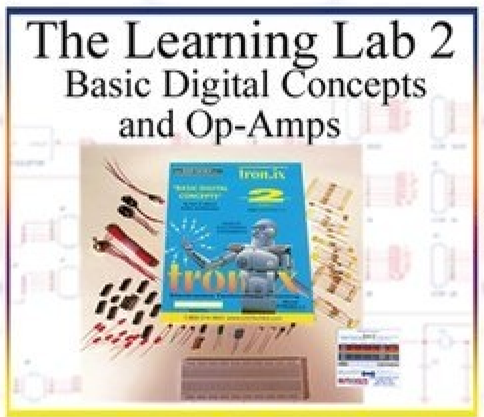
Pick a circuit!
With one PCB you have the option of detecting wirelessly: temperature, vibration, light, sound, motion, normally open switch, normally closed switch, any varying resistor input, voltage input, mA input, and tilt, just to name a few.

$32.95

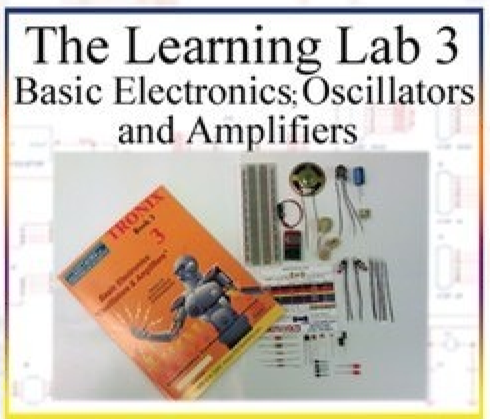
FOR BEGINNER GEEKS!



$59.95



$49.95



$39.95

These labs from LF Components show simple and interesting experiments and lessons, all done on a solderless circuit board.

As you do each experiment, you learn how basic components work in a circuit, and continue to build your arsenal of knowledge with each successive experiment.

For more info and lab details, please visit our webstore.

FEEDBACK

Continued from page 7

Beyond the Flaws

The March 2015 article on "Beyond the Arduino" has several flaws. The use of AVR studio means no chipKIT boards. "Physical Form: ..." is highly misleading as the Arduino comes in all sizes from the Trinket to the Max32. Some chipKIT chips contain RTCs, and some other chips have built-in EPROM memory.

It makes a lot of sense to put a factory built "chip" into a product because testing and debugging (which costs a lot of labor) is made way simpler. Upgrades or repairs in the field are also much faster because many times the Arduino board can be replaced without removing any wires.

Ben Heck made a pinball machine brain using a Max32 chipKIT which he just pressed onto a set of header pins on his buffer I/O board. He programs in C++ at times but chose this pre-built board.

The main problem with Arduino is the slow digital I/O. A simple port *read* or *write* fixes that. Other I/O is handled by libraries, which are likely to be written in C++ anyway. You do not have to go to a hand-built board to use C++.

chipKIT provides two development systems for their boards which, by the way, run 2.5 times faster (tested) than AVR-based boards. There is also a free Microsoft compiler with two Arduino drop-in products that ease programming.

Classes are not everything. They have overhead and they can take up more memory if they are reused instead of nesting functions by hand.

Dale Freye

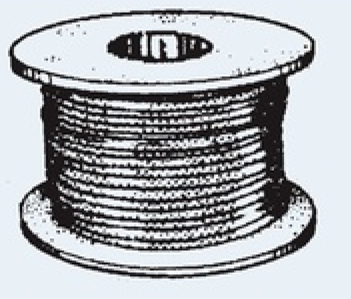
Thanks for taking the time to put your thoughts down into an email, and don't worry about me feeling bad. I appreciate your perspective! You are right, of course. There are a number of ways to tackle embedded systems, and certainly chipKIT is one of those, although I have relatively little experience with them. In working on this series, I wanted to pick a line through what can be a minefield of personal preferences and strong loyalties to certain microcontroller families, manufacturers, implementations, and even operating systems. It's a tough balancing act! I wholeheartedly agree with you that there are alternatives to what I'm trying to achieve. Which of these is best is (in my experience) usually dictated by a combination of personal preference and the specific project.

Continued on page 79

CLASSIFIEDS

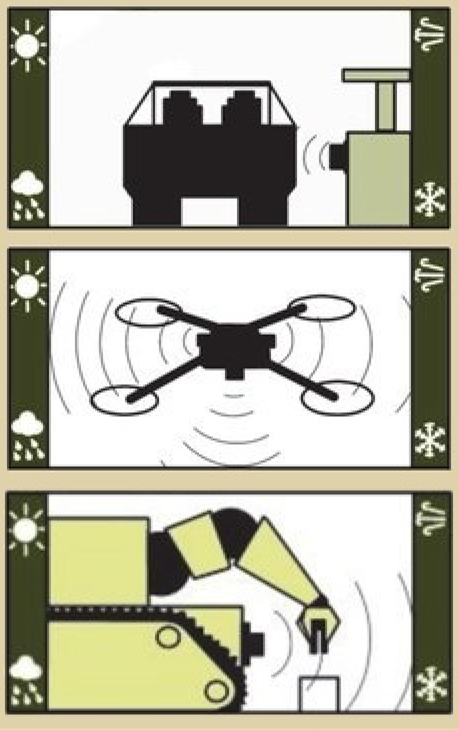
SURPLUS

WIRE/CABLE

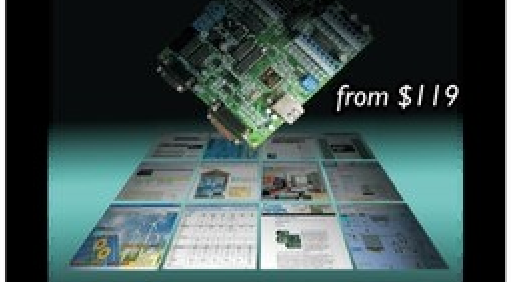


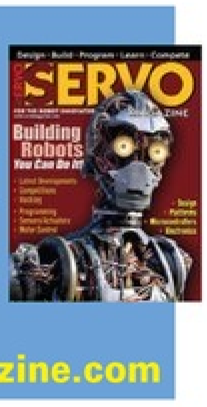
KITS/PLANS

ROBOTICS



CONTROLLERS





LIGHTING

SERVICES

HARDWARE WANTED

READER-TO-READER TECHFORUM

All New TECHFORUM Online At www.nutsvolts.com/tech-forum

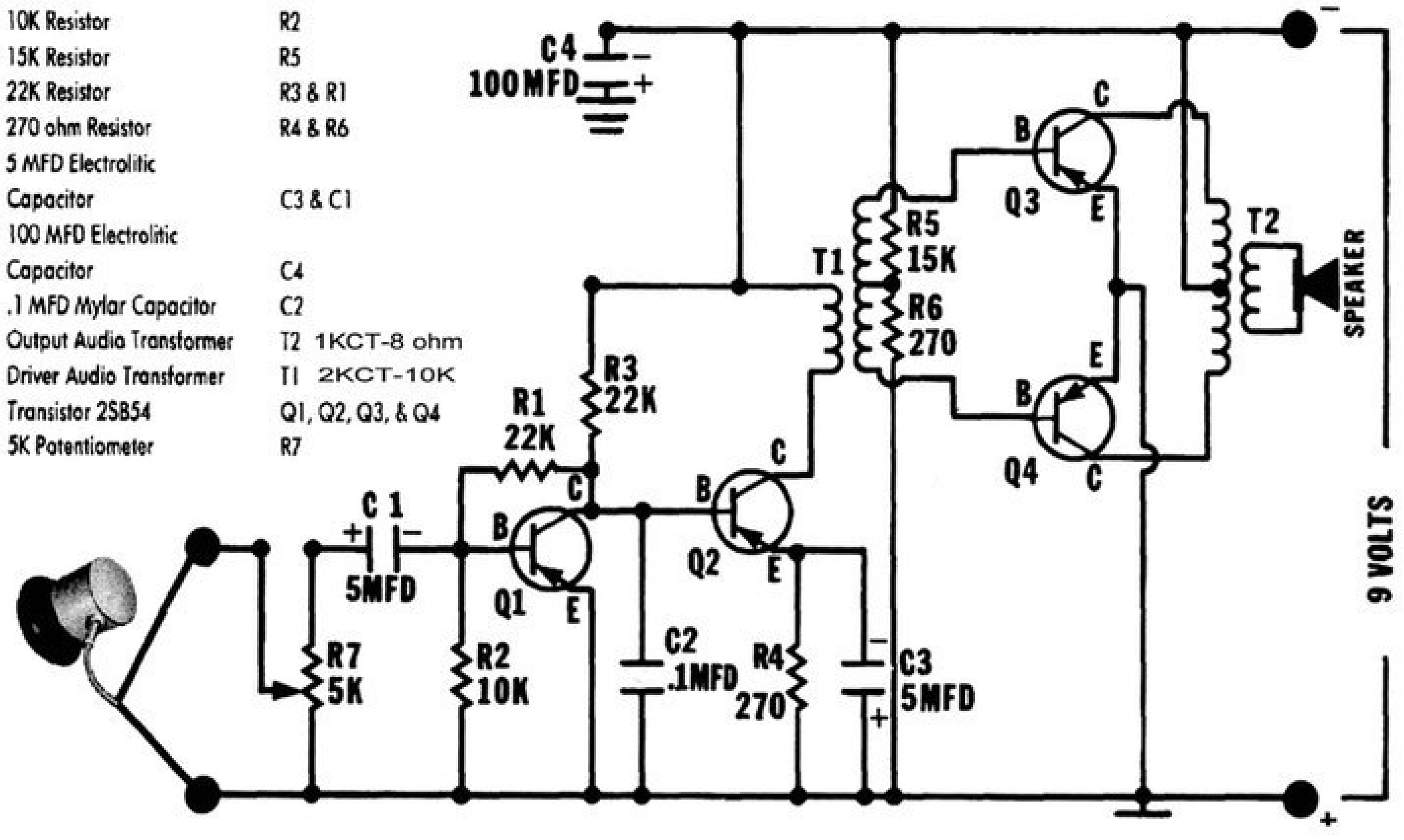
>>> QUESTIONS

Replacement Plastic Transistors

Can one of your experienced electronic engineers recommend replacement plastic transistors (maybe a la PN2907) which are easily available, in order to substitute Q1, Q2, Q3, and Q4? I want to totally eliminate 2SB54 transistors WITHOUT any draconic and/or extreme changes to the original diagram (below) for this (push-pull) phone amplifier I'm using, by attaching a suction cup to my (land-line) phone's receiver.

I'm well aware of the existence of other phone amplifiers (built around the LM386) which are easier to build with fewer parts, but I'm only interested in this diagram, as I've already built it in the past — not only as a phone amp with ample volume (which also works very well as a P.A. system), but as a wired intercom.

#5151 **Nate Franklin**
Schererville, IN



10K Resistor	R2
15K Resistor	R5
22K Resistor	R3 & R1
270 ohm Resistor	R4 & R6
5 MFD Electrolitic Capacitor	C3 & C1
100 MFD Electrolitic Capacitor	C4
.1 MFD Mylar Capacitor	C2
Output Audio Transformer	T2 1KCT-8 ohm
Driver Audio Transformer	T1 2KCT-10K
Transistor 2SB54	Q1, Q2, Q3, & Q4
5K Potentiometer	R7

Voltage Reduction

What would be the most efficient way to reduce the voltage from a nine volt battery to 5V? I could use a 7805 but it seems to bleed off a lot of power as heat. Is there a more efficient circuit or part?

#5152 **Alan Vallo**
Piedmont, SC

On/Off Circuit

Is there a simple circuit that would allow a normally open pushbutton to turn on a relay when pressed, and then turn it off when pressed again?

#5153 **Isaac Wright**
Maumee, OH

Fuse Confusion

I have a drill press that has a five amp fuse that recently has started to blow on just about every project. I am not working on harder material and the bits are sharp, so I don't think its increased torque from friction which leaves the electronics. As this is simply a motor and a power switch, I can't imagine what might be causing the increased current draw. Suggestions?

#5154 **Liam Olivo**
Saginaw, MI

All questions *AND* answers are submitted by *Nuts & Volts* readers and are intended to promote the exchange of ideas and provide assistance for solving technical problems. All submissions are subject to editing and will be published on a space available basis if deemed suitable by the publisher. Answers are submitted by readers and **NO GUARANTEES WHATSOEVER** are made by the publisher. The implementation of any answer printed in this column may require varying degrees of technical experience and should only be attempted by qualified individuals.

Always use common sense and good judgment!

>>> ANSWERS

[#4151 - April 2015]
LED vs. Incandescent Lamps

I have been tasked with the chore of replacing 300 watt incandescents (5,900 lumen). How many of what kind of LEDs and current-limiting diodes in series/parallel do I need to fool the human eye into thinking it is seeing a brighter, more pleasant level of lumens?

I recently replaced twelve 300 watt incandescent lamps with LED floods. In my case, it was made easier because the 300 watt lamps illuminated hanging lamp enclosures in a church. The overall purpose was to provide reading illumination for the congregants.

If this application suits your purpose, look at Lighting Science DFN38WWV2NFL120, which is a 120V 24 watt PAR38 lamp, 3000K. These lamps — while only rated at 1,300 lumens — put all of their light output in one direction (downwards), while the incandescent lamps spread their 5,900 lumens all over the room. Plus, they're dimmable. It was a no-brainer for us because they worked in the application and the electrical demand for the lighting went from 3,600 watts to 288 watts.

One on-line source is **https://www.1000bulbs.com/product/63140/LED-PAR38242530.html**, but your local electrical supply jobber might get you a better price — especially if your state is currently giving rebates for so-called "green" appliances.

Peter A. Goodwin
Rockport, MA

Continued from page 77

I've tried to stick with the heart of the Arduino — the ATmega328P with all of its limitations and benefits — and keep the concepts as true to the microcontroller as possible. By doing this, I hope that readers may more easily transition to working with raw microcontrollers, learn from the path that I struggled along, and be able to put the learning to use in exploring the broader options out there.

We've got much ground to cover, and I hope that you'll keep reading and keep the ideas and comments coming. Please do drop me a line with any thoughts — I enjoy these kinds of discussions as they broaden horizons all round.

Andrew Retallack

"Beyond" Fan

I just wanted to thank Andrew Retallack for his article "Beyond the Arduino." As someone who prefers individual components in projects rather than reusing a new development board each time (e.g., MSP430 vs. Launchpad; PIC16F84 vs. BASIC Stamp; etc.), this was a refreshing change of pace. With little to go on in the way of designing with ATmega ICs instead of an Arduino board, maybe now I too can try my hand at using this little processor in a few of my own projects.

Maybe others can relegate the Arduino back to the task for which it is best suited — a reusable prototype board for developing projects which can then be migrated to their own ATmega-based circuits.

Derek Tombrello

Propeller Preference

The frequency counter article by Jim Teixeira in the March issue is similar to something I have been planning to replace; something I built about 40 years ago.

However, I would suggest replacing most of the hardware and the PIC micoprocessor with a Parallax Propeller. It will accept inputs greater than 20 MHz, and its eight processors each have two built-in 32-bit multipurpose counters — along with the clock counter — that will provide all of the needed counting functions (and a lot more, if needed). Programming these functions is almost (but not quite) trivial.

Jerry Nicholson

PICAXEd Apart

Though I'm not involved in the PICAXE platform, I've been following Ron Hackett's PICAXE Primer column lately. Although the circuits probably end up being functional, his explanation of transistor basics couldn't be more wrong. I refer to the "explanation" of the functioning of an NPN transistor.

Since the articles are targeting novice hobbyists, it's important to get the fundamentals right, rather than causing confusion.

1. There are TWO junctions in a bipolar transistor, not three. They are the base-emitter junction and the collector-base junction. Perhaps you meant "three terminals."

"The switch will turn on (i.e., current will flow from the collector to the emitter) whenever the base is 'taken toward' (whatever that means) the collector, i.e., whenever the base is connected to the same voltage level as the collector."

2. A bipolar transistor is a current operated device, not voltage, i.e., the collector voltage has little control over how much current flows from the collector to the base. Current flowing into the base-emitter junction permits current to flow from the collector to the emitter, given almost any collector voltage.

The ultimate collector voltage is dependent on the load resistor and the collector CURRENT, V = IR. Approximately, Ice = β * Ibe — the collector current is equal to the base current times the beta (or current gain) of the transistor.

In order for this equation to be met, there must be limited current flowing into the base which, in general terms, requires the voltage of the base to be above approx. 0.6V for silicon. Also, the collector voltage must be above zero. The equation sets the upper limit for collector current once Vc is above Vce(sat); typically, a few tenths of a volt under these operating conditions.

Later in his article, "When the base of the NPN transistor is at +5, the collector is (almost) at ground." If the base were at +5, the transistor would act as a fuse, shorting the B-E junction and melting the base bond wire. He qualifies it earlier by stating R1 limits current, but still puts the base at +5 — which it is not. Maybe you should measure Vbe.

I'm sure a tutorial on the operation of a bipolar transistor or circuit is not necessary, having built many successful projects, and smoked his share of parts (as have I). However, I feel this project is full of misconceptions.

I honestly don't understand why Mr. Hackett chose not to use a CMOS hex inverter for the adapter. It's simpler, uses fewer components, saves power, and the operation of FETs is easier to explain in voltage terms than a BJT. It is certainly as space efficient as two transistors and six resistors. Even using the stripboard technique, this project could be built one row narrower and shorter. Or, not even bother with hardware and do it all in firmware with the XOR instruction.

Steve McChrystal

Thanks for your feedback, Steve, on the March 2015 PICAXE Primer Column. It has certainly made me aware of the fact that I should have been more careful (and more precise) in my explanation of the functioning *of an NPN transistor when it's used as a switch. The following are my comments on the points you raised.*

Of course, I know that bipolar transistors contain two junctions, not three. You're correct. I meant to say "terminals," and somehow that error eluded me, even though I re-read the article several times before submitting it.

I don't understand your addition of "whatever that means," when I clearly specified what I meant in the same sentence.

However, I do think I could have chosen my wording more carefully.

Yes, I do know that a transistor is a current-operated device. In fact, I know it so well, that I forgot to mention it, so your point is well-taken. I think my omission may have confused more than a few readers.

The statement that you quote ["When the base of the NPN transistor is at +5V, the collector is (almost) at ground."] is the one that concerns me the most. What I meant (and certainly should have said) is "When the base of the NPN transistor is tied to +5V via the R1 current-limiting resistor, the collector is (almost) at ground."

I honestly don't know how I managed to let that one slip by. In fact, I did measure Vbe at the time, and it was about +0.7V when the input to R1 was +5V, so I certainly should have spotted my own mis-statement.

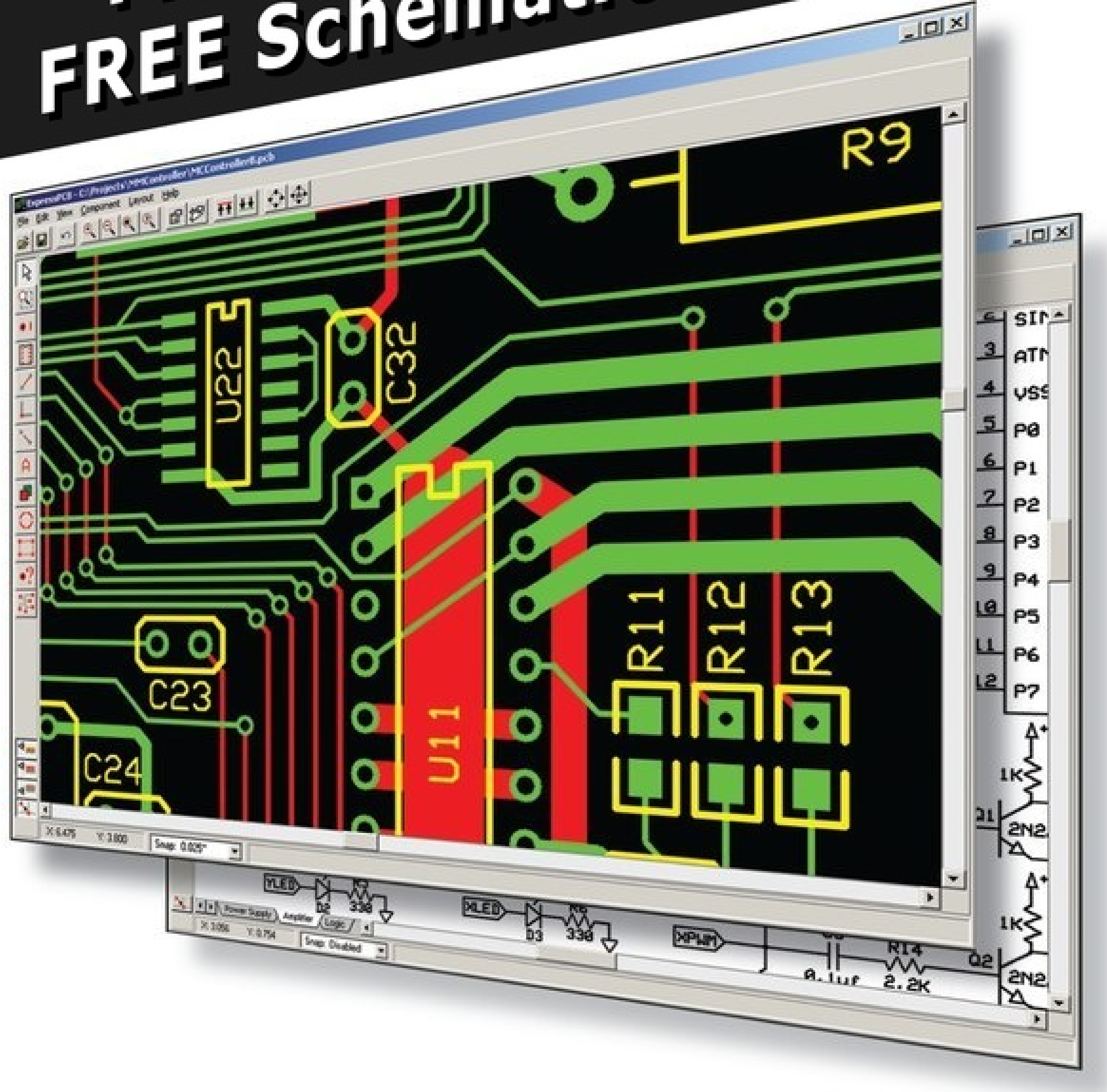
I do agree that logic inverters would be simpler to explain, but I chose to use transistors because I believe more of my readers would have NPN transistors on hand (rather than logic inverters).

However, I did use logic inversion (two NAND gates rather than a hex inverter) for a PCB version of the same project, but that approach did not turn out to be more space-efficient than the transistors because a few resistors were still needed to protect against accidental misconnections.

(Unfortunately, a firmware solution isn't possible for the PICAXE programming connections because the PICAXE firmware is not accessible to the user.)

Finally, I want to thank you again for your feedback. It wasn't easy to hear, but it certainly made me realize that I need to more carefully scrutinize my own writing, to be sure that it clearly conveys what I intend, and that it accurately explains the topic at hand.

Ron Hackett



■LOOK FOR ■SEARCH FOR ■FIND

Find your favorite advertisers here!

ADvertiser INDEX

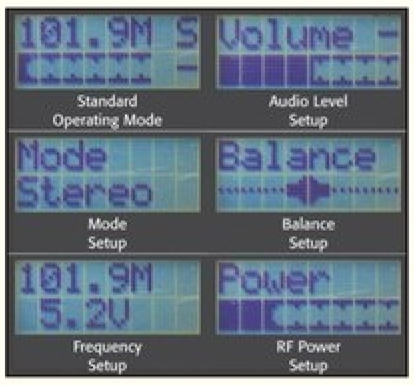
It's Time To Build A Kit!

We Put The FUN In Electronics!

Digital Controlled FM Stereo Transmitters

- ✔ PLL synthesized for drift free operation
- ✔ Front panel digital control and display of all settings and parameters!
- ✔ Professional metal case for noise-free operation
- ✔ EMI filtering on audio and power inputs
- ✔ Super audio quality, rivals commercial broadcasts
- ✔ Available in domestic kit or factory assembled export versions

Our Most Popular Transmitter... On Sale NOW!



For more than a decade we've been the leader in hobbyist FM radio transmitters. We told our engineers we wanted a new technology transmitter that would provide FM100 series quality without the advanced mixer features. They took it as a challenge and designed not one, but TWO transmitters!

The FM30B is designed using through-hole technology and components and is available only as a do-it-yourself kit with a 25mW output very similar to our FM25 series. Then the engineers redesigned their brand-new design using surface mount technology (SMT) for a very special factory assembled and tested FM35BWT version with 1W output for our export only market!

All settings can be changed without taking the cover off! Enter the setup mode from the front panel and step through the menu to make all of your adjustments. A two line LCD display shows you all the settings! In addition to the LCD display, a front panel LED indicates PLL lock so you know you are transmitting.

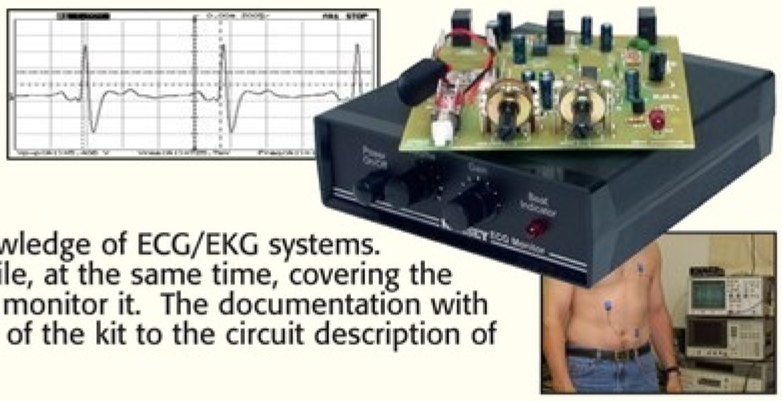
Besides frequency selection, front panel control and display gives you 256 steps of audio volume (left and right combined) as well as RF output power. A separate balance setting compensates for left/right differences in audio level. In addition to settings, the LCD display shows you "Quality of Signal" to help you set your levels for optimum sound quality. And of course, all settings are stored in non-volatile memory for future use! Both the FM30B and FM35BWT operate on 13.8 to 16VDC and include a 15VDC plug-in power supply. The stylish black metal case measures 5.55"W x 6.45"D x 1.5"H.

Take advantage of our ***Spring Sale***, and get our FMA200E 3.4dB gain FM antenna below to maximize your system while also saving big bucks!

(Note: After assembly of this do-it-yourself hobby kit, the user is responsible for complying with all FCC rules & regulations within the US, or any regulations of their respective governing body. FM35BWT is for export use and can only be shipped to locations outside the continental US or valid APO/FPO addresses or valid customs brokers for end delivery outside the continental US.)

FM30B	**Digital FM Stereo Transmitter Kit, 0-25mW, Black**	SPRING SALE! ***$169.95***
FM35BWT	**Digital FM Stereo Transmitter, Assembled, 0-1W, Black (Export ONLY)**	***$259.95***

Electrocardiogram ECG Heart Monitor



- ✔ Visible and audible display of your heart rhythm!
- ✔ Bright LED "Beat" indicator for easy viewing!
- ✔ Re-usable hospital grade sensors included!
- ✔ Monitor output for professional scope display
- ✔ Simple and safe 9V battery operation

Use the ECG1C to astound your physician with your knowledge of ECG/EKG systems. Enjoy learning about the inner workings of the heart while, at the same time, covering the stage-by-stage electronic circuit theory used in the kit to monitor it. The documentation with the ECG1C covers everything from the circuit description of the kit to the circuit description of the heart!

Multiple "beat" indicators include a bright front panel LED that flashes with the actions of the heart along with an adjustable level audio speaker output that supports both mono and stereo hook-ups. In addition, a monitor output is provided to connect to any standard oscilloscope to view the traditional style ECG/EKG waveforms just like you see on ER... or in the ER! 10 hospital grade re-usable probe patches are included together with the matching custom case set shown. Safe 9V battery operation.

ECG1C	**Electrocardiogram Heart Monitor Kit With Case & Patches**	**$44.95**
ECG1WT	**Electrocardiogram Heart Monitor, Factory Assembled & Tested**	**$89.95**
ECGP10	**Electrocardiogram Re-Usable Probe Patches, 10-Pack**	**$4.95**

Passive Aircraft Monitor

PATENTED!

The hit of the decade! Our patented receiver hears the entire aircraft band without any tuning! Passive design has no LO, therefore can be used on board aircraft! Perfect for airshows, hears the active traffic as it happens! Available kit or factory assembled.

ABM1	**Passive Aircraft Receiver Kit**	**$89.95**

Laser Trip Sensor Alarm

True laser protects over 500 yards! At last within the reach of the hobbyist, this neat kit uses a standard laser pointer (included) to provide both audible and visual alert of a broken path. 5A relay makes it simple to interface! Breakaway board to separate sections.

LTS1	**Laser Trip Sensor Alarm Kit**	**$29.95**

Collinear Vertical FM Antenna

Our 5/8 wave omni antenna has been the standard for LPFM insallations worldwide. Provides 3.4dB gain while keeping the signal radiation low to the horizon for maximum range. Field tuneable over the entire FM range for a perfect match. SO239 connector for PL259 plug.

FMA200E	**Omnidirectional FM Antenna**	SPRING SALE! ***$119.95***

Logic Interface Module

Interface your digital output to the real world with an on-board SPDT relay rated at 240V at 10A! It takes a digital low (.5VDC or less) or a high (+1 to +12VDC) and provides your choice of an active low or high closure! It's that simple! Runs on 12VDC at 60mA.

RI1	**Logic Interface Kit**	**$17.95**

Four-Mode Keyless Entry Test Set

Just like the days of "plugs, points, and condenser" are over, so are the days of having the hardware store grind out a spare key for your car!

Now when your keyless access system doesn't work, you need to accurately detect what part of the system is malfunctioning. This could be anything from a dead battery in the key fob, a "brain-dead" key fob, to malfunctioning sensors, antennas, or other system components in the vehicle. Until now there was no way to determine where the system was failing.

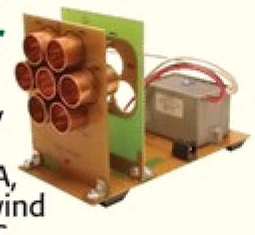
Testing your system is easy. To test the complete 125 kHz/315 MHz communications path just stand close to the vehicle with the WCT3 and your key fob in hand. Press the test button and the WCT3 will detect and display the presence of the vehicle's 125kHz/20KHz signal and, if they "handshake", will also detect and display the presence of your key fob's 315MHz return signal. You can independently test key fob only signals (panic, lock, trunk, etc.) by holding the key fob near the WCT3, pressing the test button, and pushing the function button on the key fob.

The same functionality testing can be done with IR key fobs. The modulated IR signal is detected and will illuminate the IR test LED on the test set.

If you know a few "secrets" you can also see if the tire pressure sensors/transmitters are generating signals or the built-in garage door opener in your rear view mirror is transmitting a signal! Runs on a standard 9V battery. Also available factory assembled & tested.

WCT3	**Keyless Entry Test Set Kit**	**$59.95**

Air Blasting Ion Generator



Generates negative ions along with a hefty blast of fresh air, all without any noise! The steady state DC voltage generates 7.5kV DC negative at 400uA, and that's LOTS of ions! Includes 7 wind tubes for max air! Runs on 12-15VDC.

IG7	**Ion Generator Kit**	**$64.95**

HV Plasma Generator

Generate 2" sparks to a handheld screwdriver! Light fluorescent tubes without wires! This plasma generator creates up to 25kV at 20kHz from a solid state circuit! Build plasma bulbs from regular bulbs and more! Runs on 16VAC or 5-24VDC.

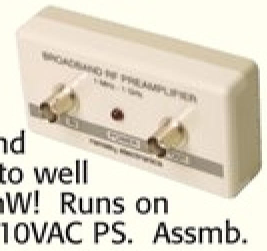
PG13	**HV Plasma Generator Kit**	**$64.95**

Speedy Speed Radar Gun

Our famous *Speedy* radar gun teaches you doppler effect the fun way! Digital readout displays in MPH, KPH, or FPS. You supply two coffee cans! Runs on 12VDC or our AC121 supply.

SG7	**Speed Radar Gun Kit**	**$74.95**

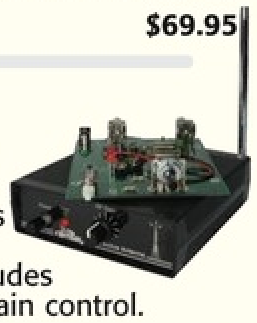
Broadband RF Preamp



Need to "perk-up" your counter or other equipment to read weak signals? This preamp has low noise and yet provides 25dB gain from 1MHz to well over 1GHz. Output can reach 100mW! Runs on 12 volts AC or DC or the included 110VAC PS. Assmb.

PR2	**Broadband RF Preamp**	**$69.95**

Active Receive Antenna



The popular antenna for the serious DX'ers works on all bands - shortwave, HF, VHF, and UHF yet performs like a 60' long wire antenna! Provides over 15dB of gain, and includes auto-off RF bypass and front panel gain control.

AA7C	**Active Antenna Kit**	**$59.95**

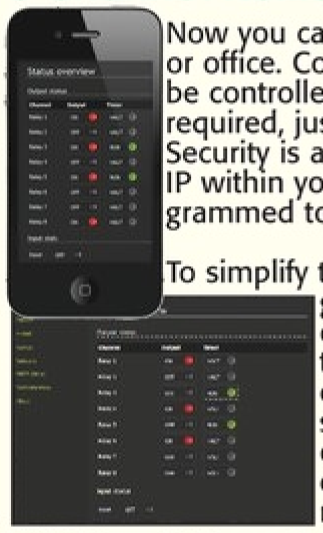
ramsey
www.ramseykits.com

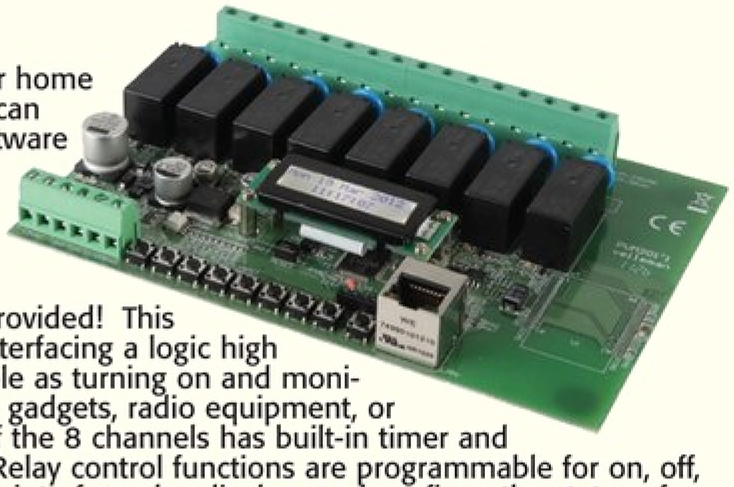
There's only so much room on these two pages, so check it all out in our new virtual electronic catalog! Flip through the pages and search with ease! Visit ***www.ramseycatalog.com***

Follow Us and SAVE $$

Follow us on your favorite network site and look for a lot of super deals posted frequently... exclusively for our followers!

8-Channel Remote Ethernet Controller



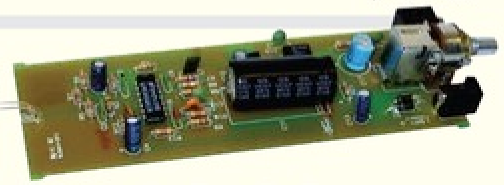


Now you can easily control and monitor up to 8 separate circuits via the standard Ethernet network in your home or office. Connection wise it couldn't be simpler. The controller functions as an IP based web server, so it can be controlled by any internet browser that can reach your network! There are no drivers or proprietary software required, just access the controller like any web page from your PC, laptop, or even your smartphone! Security is assured allowing up to 4 separate user credentials. The controller can be set to a specific static IP within your network subnet or can be set to DHCP (auto negotiate). The controller can even be programmed to send you an email to notify and confirm power up and status changes!

To simplify the connection of your equipment to the controller, 8 separate and isolated relay outputs are provided! This gives you internet or network control of up to 8 separate functions. No need to worry about interfacing a logic high or logic low, or burning up the interface! The applications are endless! From something as simple as turning on and monitoring lights at your house with a normal latched closure to advanced control of your electronic gadgets, radio equipment, or even your garage door! Each relay contact is rated at 12A at 30VDC or 16A at 230VAC. Each of the 8 channels has built-in timer and scheduler programs for day, weekend, working days, every day, and every day except Sunday. Relay control functions are programmable for on, off, or pulse (1-99 seconds, 1-99 minutes, or 1-99 hours). In addition to control functions, the web interface also displays and confirms the status of each channel. Each channel can be custom labeled to your specific function name. The controller operates on 12VDC or 12VAC at 500mA or our new AC121 global 12VDC switching power supply below. Factory assembled, tested, and ready to go! Even includes a Cat-5 cable!

VM201 8-Channel Remote Ethernet Controller, Factory Assembled & Tested $169.95

Laser Beam Audio Communicator



Now you can talk to your friends over one of the most secure long-distance transmission types available, a laser beam! The transmitter uses a microphone or external audio to modulate a laser beam on and off at a rate of more than 16kHz so the audio fidelity is much better than that of a standard 3kHz telephone line! And being an optical path, there certainly won't be any 3-letter agencies or anyone else listening!

The receiver includes filtering to remove the 16kHz carrier and leave behind the high quality audio, and then boost its level for use with earphones. Transmitter audio AGC keeps your level perfect! Includes transmitter, receiver and laser pointer. Each runs on a 9V battery (not included).

LBC6K Laser Beam Audio Communicator Kit $59.95

Signal Magnet Antenna

The impossible AM radio antenna that pulls in the stations and removes the noise, interference, and static crashes from your radio! Also helps that pesky HD AM Radio stay locked! Also available factory assembled.

SM100 Signal Magnet Antenna Kit $89.95

Tickle-Stick Shocker

The kit has a pulsing 80 volt tickle output and a mischievous blinking LED. And who can resist a blinking light and an unlabeled switch! Great fun for your desk, "Hey, I told you not to touch!" Runs on 3-6 VDC.

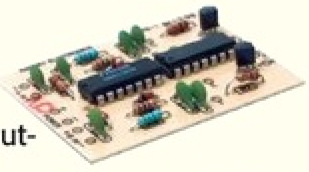
TS4 Tickle Stick Kit $9.95

Water Sensor Alarm

This little $7 kit can really "bail you out"! Simply mount the alarm where you want to detect water level problems (sump pump)! When the water touches the contacts the alarm goes off! Sensor can even be remotely located. Runs on a standard 9V battery.

MK108 Water Sensor Alarm Kit $6.95

Touch Switch



The ultimate touch switch! Touch once - it's on, touch again - it's off, or use the momentary outputs that stay on only as long as touched. Two switch circuits on each board. Drives loads up to 100mA. Runs on 6-12VDC.

TS1 Touch Switch Kit $9.95

Tri-Field Meter Kit

THE GHOST DETECTOR!

"See" electrical, magnetic, and RF fields as a graphical LED display on the front panel! Use it to detect these fields in your house, find RF sources, you name it. Featured on CBS's Ghost Whisperer to detect the presence of ghosts! Req's 4 AAA batteries.

TFM3C Tri-Field Meter Kit $74.95

Electret Condenser Mic

This extremely sensitive 3/8" mic has a built-in FET preamplifier! It's a great replacement mic, or a perfect answer to add a mic to your project. Powered by 3-15VDC, and we even include coupling cap and a current limiting resistor! Extremely popular!

MC1 Mini Electret Condenser Mic Kit $3.95

Audio Recorder & Player

Record and playback up to 8 minutes of messages from this little board! Built-in condenser mic plus line input, line & speaker outputs. Adjustable sample rate for recording quality. 4-switch operation that can be remote controlled! Runs on 9-12VDC at 500mA.

K8094 Audio Recorder/Player Kit $32.95

Laser Light Show

Just like the big concerts, you can impress your friends with your own laser light show! Audio input modulates the laser display to your favorite music! Adjustable pattern & speed. Runs on 6-12VDC.

LLS1 Laser Light Show Kit $49.95

12VDC Regulated Switching Supply

Go green with our new 12VDC 1A regulated supply. Worldwide input 100-240VAC with a Level-V efficiency! It gets even better, includes DUAL ferrite cores for RF and EMI suppression. All this at a 10 buck old wallwart price! What a deal!

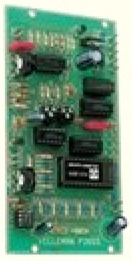
AC121 12VDC 1A Regulated Supply $9.95

12VDC Worldwide Supply

It gets even better than our AC121 above! Now, take the regulated Level-V green supply, bump the current up to 1.25A, and include multiple blades for global country compatibility! Dual ferrite cores!

PS29 12VDC 1.25A Global Power Supply $19.95

Electronic Watch Dog



A barking dog on a PC board! And you don't have to feed it! Generates 2 different selectable barking dog sounds. Plus a built-in mic senses noise and can be set to bark when it hears it! Adjustable sensitivity! Unlike my Greyhound, it eats 9-12VDC, it's not fussy!

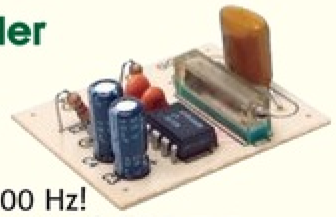
K2655 Electronic Watch Dog Kit $39.95

Sniff-It RF Detector Probe

Measure RF with your standard DMM or VOM! This extremely sensitive RF detector probe connects to any voltmeter and allows you to measure RF from 100kHz to over 1GHz! So sensitive it can be used as a RF field strength meter!

RF1 Sniff-It RF Detector Probe Kit $27.95

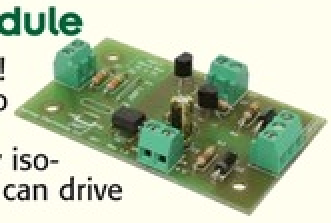
Tone Encoder/Decoder



Encode and decode with the same kit! This little mini-kit will simultaneously encode and/or decode any audio frequency between 40Hz and 5,000 Hz! Precision 20-turn trim pot adjustment! 5-12VDC.

TD1 Tone Encoder/Decoder Kit $9.95

Optically Isolated Module



The hobbyist's headache solver! Converts any AC or DC signal to logic level. The beauty is that the input and output are totally isolated from each other! Output can drive up to 150mA at 40VDC.

OM2 Optically Isolated Module Kit $16.95

Classic Nixie Tube Clocks

HOT SELLER!

Our next generation of classic Nixie tube clocks perfectly mesh today's technology with the Nixie era technology of the 60's. Of course, features you'd expect with a typical clock are all supported with the Nixie clock... and a whole lot more!

The clocks are programmable for 12 or 24 hour mode, various AM/PM indications, programmable leading zero blanking, and include a programmable alarm with snooze as well as date display, 4 or 6 tube, kit or assembled!

We then jumped the technological time line of the 60's Nixie displays by adding the latest multi-colored LEDs to the base of the Nixie tubes to provide hundreds of illumination colors to highlight the glass tubes! The LED lighting can be programmed to any color and brightness combination of the colors red, green, or blue to suit your mood or environment.

Then we leaped over the technological time line by integrating an optional GPS time base reference for the ultimate in clock accuracy! The small optional GPS receiver module is factory assembled and tested, and plugs directly into the back of the clock to give your Nixie clock accuracy you could only dream of!

The clocks are available in our signature hand rubbed Teak & Maple or clear acrylic bases. And for the 6-tube clocks you also have your choice of IN-14 nixie tubes (5=upside down 2!) or highly sought after IN-8-2 nixie tubes.

NIXIE Classic Nixie Tube Clock Kits From $229.95

GET THE NUTS AND VOLTS DISCOUNT!

*Mention or enter the coupon code **NVRMZ142** and receive 10% off your order!*

800-446-2295

www.ramseykits.com

Prices, availability, and specifications are subject to change. We are not responsible for typos, stupids, printer's bleed, or upcoming summer dreams that just aren't materializing! But unlike Phil, Robin says early spring and nice summer, so there is hope! Visit www.ramseykits.com for the latest pricing, specials, terms and conditions. Copyright 2015 Ramsey Electronics®... so there!

RAMSEY ELECTRONICS®
590 Fishers Station Drive
Victor, NY 14564
(800) 446-2295
(585) 924-4560

SIGLENT

5% EDUCATIONAL DISCOUNT!

THE BEST VALUE IN TEST EQUIPMENT

$319

SDS1072CML OSCILLOSCOPE

70MHz, 2 ch, unique EasyHunting™ waveform capture. 1GSa/s 2Mpt memory, built-in 6 digit counter.

$248

SDG805 AWG GENERATOR

5MHz, 1 ch, 125MSa/s EasyPulse™ technology for low jitter, fast rising/falling edges at every frequency.

$1489

SDS2104 OSCILLOSCOPE

100MHz, 4 ch, 2GSa/s, 28Mpt memory, 110,000 wfm/s capture, huge 8” color LCD, AWG/Logic analyzer options.

$465

SHS806 HANDHELD OSCILLOSCOPE

60MHz, 1GSa/s, 2Mpt memory portable scope with features from expensive bench-top instruments.

$369

SPD3303D POWER SUPPLY

High precision 195 watt supply, 3 outputs: 30V@3A x2, 2.5-5V@3A, 4 digit voltage / 3 digit current display.

$469

SDM3055 MULTIMETER

5 $^{1/2}$ digit, 150 readings/sec with math and statistical functions and remote communication interfaces.

INCREDIBLE LOW PRICES, EXCELLENT CUSTOMER SERVICE, FREE TECH SUPPORT

(585) 385-1750 WWW.SAELIG.COM INFO@SAELIG.COM

Saelig .com
unique electronics